НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації і управління

До захисту допущено:
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім'я, прізвище)
"___"_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «*Інформаційна система технології менеджменту проектів з використанням Rest API* »

Виконав: студент *IV* курсу, групи *ІС-61*

Фомін Ілля Дмитрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник *ст.викл. Новікова Поліна Анатоліївна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Консультант з графічної документації *ст.викл. Проскура Світлана Леонідівна* _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент Доц. каф. ТК Корнага Ярослав Ігорович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (вл.ім’я, прізвище)
“___”_____2020 р.

З А В Д А Н Н Я
на дипломний проєкт студенту

Фоміну Іллі Дмитрувачу
(прізвище, ім’я, по батькові)

1. Тема проєкту «Інформаційна система технології менеджменту проектів з використанням Rest API »

керівник проєкту *Новікова Поліна Анатоліївна, ст.викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “*7*”*травня* 2020 р. №1081-с

2. Термін подання студентом проєкту “*01*”*червня* 2020 року

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв’язання

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1. Схема структурна варіантів використань

2. Схема структурна класів програмного забеспечення

3. Схема бази даних

4. Схема структурна послідовності

5. Схема структурна послідовності

6. Схема структурна послідовності

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«13» квітня 2020 року*

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*18.04.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*21.04.2020*	
3.	*Постановка та формалізація задачі*	*24.04.2020*	
4.	*Розробка інформаційного забезпечення*	*30.04.2020*	
5.	*Алгоритмізація задачі*	*06.05.2020*	
6.	*Обґрунтування використовуваних технічних засобів*	*11.05.2020*	
7.	*Розробка програмного забезпечення*	*21.05.2020*	
8.	*Налагодження програми*	*26.05.2020*	
9.	*Виконання графічних документів*	*28.05.2020*	
10.	*Оформлення пояснювальної записки*	*29.05.2020*	
11.	*Подання ДП на попередній захист*	*15.05.2020*	
12.	*Подання ДП на основний захист*	*01.06.2020*	
13.	*Подання ДП рецензенту*	*02.06.2020*	

Студент Ілля ФОМІН

Керівник Поліна НОВІКОВА

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6326.00.000 ПЗ	Пояснювальна записка	133	
3	А4	ДП 6326.01.000 ТЗ	Технічне завдання	9	
4	А3	ДП 6326.02.000 ССВ	Схема структурна варіантів використань	1	
5	А3	ДП 6326.03.000 ССК	Схема структурна класів програмного	1	
			забеспечення		
6	А3	ДП 6326.04.000 СБД	Схема бази даних	1	
7	А3	ДП 6326.05.000 ССП	Схема структурна послідовності	1	
8	А3	ДП 6326.06.000 ССП	Схема структурна послідовності	1	
9	А3	ДП 6326.07.000 ССП	Схема структурна послідовності	1	

Зм.	Арк.	ПІБ	Підп.	Дата	ДП 6326 00.000.00		
Розробн.		Фомін І.Д.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Новікова П.А.					
Консульт.						КПІ ім. Ігоря Сікорського Каф. АСОІУ Гр. ІС-63	
Н/контр.		Проскура С.Л.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проєкту

на тему: Інформаційна система технології менеджменту проектів з використанням Rest API

Київ – 2020 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 19 рисунків, 19 таблиць, 1 додаток, 6 джерел.

Дипломний проект присвячений розробці комплексу задач з автоматизації менеджменту виконання проектів та оцінювання продуктивності учасників системи.

У розділі інформаційного забезпечення приведено опис предметного середовища, огляд наявних аналогів та постановка задачі.

Розділ математичного забезпечення присвячений побудові рейтингової оцінки учасника системи на основі виконаних ним завдань.

Програмне забезпечення, будучи клієнт-серверним застосунком, надає змогу автоматично вирішувати який з користувачів системи здатен виконати поставлене завдання.

У технологічному розділі знаходиться перелік тестів та керівництво користувача.

ІНФОРМАЦІЙНІ СИСТЕМИ, МЕНЕДЖМЕНТ ПРОЕКТІВ, АВТОМАТИЗАЦІЯ, REST API, ВЕБ ЗАСТОСУНКИ.

ABSTRACT

Structure and scope of work. The explanatory note of the diploma project consists of six sections, contains 20 figures, 18 tables, 1 appendix, 6 sources.

The diploma project is devoted to the development of a set of tasks for automation of project implementation management and performance evaluation of system participants.

The section of information support provides a description of the subject environment, an overview of existing analogues and problem statement.

The section of mathematical support is devoted to the construction of the rating of the system participant on the basis of the tasks performed by him.

The software, being a client-server application, allows you to automatically decide which of the users of the system is able to perform the task.

The technology section contains a list of tests and user manual.

INFORMATION SYSTEMS, PROJECT MANAGEMENT, AUTOMATION, REST API, WEB APPLICATIONS.

ЗМІСТ

ВСТУП

Дипломний проект присвячений розробці комплексу задач з автоматизації менеджменту виконання проектів та оцінювання продуктивності учасників системи, представляє собою ефективний метод контролю виконання завдань на проектах, що були зареєстровані в системі.

Розроблена система може бути використана у будь-яких галузях діяльності людини, оскільки всі процеси мають учасників, що виконують певні завдання, і кожному учаснику призначається задача згідно ролі у проекті та рівня експертизи. Для керівників різних проектів буде набагато простіше обирати виконавців різних завдань, використовуючи рейтинг учасників в розробленій системі.

У будь-якому проекті дуже важливим є правильне формування та декомпозиція задач, розроблена в рамках дипломного проекту система дозволяє оптимізувати процес роботи із цими задачами завдяки автоматизації оцінювання продуктивності учасників,.

Практичне значення одержаних результатів. Розроблена інформаційна система технології менеджменту проектів з використанням REST API.

Публікації. Результати роботи були опубліковані у тезах доповідей на науково-технічній конференції [1]

1 Загальні положення

1.1 Опис предметного середовища

Розроблена система повинна надавати змогу учасникам створювати власні проекти або під’єднуватись до вже існуючих. У випадку створення проектів учасники стають головою проекту і мають змогу передивлятись статистичну інформацію в рамках проекту, таку як загальна кількість завдань на проекті, виконані завдання, невиконані завдання, які завдання були виконані вчасно, які були прострочені, статистику окремих учасників проекту. У випадку під’єднання до проекту учасник має змогу створити завдання та задати для нього термін виконання, призначити або змінити виконавця (при зміні термін виконання не скидається), залишити коментар до свого завдання і змінювати статус завдання (виконується або не виконується), може перевірити свою статистику, а саме: кількість вже виконаних учасником завдань, які були виконані в строк, а які прострочені та від яких учасник відмовився. Учасники можуть покидати проекти, а голова може закрити проект. В такому випадку всі учасники покидають проект, однак він залишається в БД, і тільки голова може його переглянути або звільнити від участі якогось учасника.

1.1.1 Опис процесу діяльності

Користувач, авторизувавшись\зареєструвавшись, обирає або створює проект, після чого може або сам обирати завдання, або інші користувачі на проекті можуть закріпити якесь завдання користувачу. Кожен учасник на проекті може створити завдання та призначити виконавця, однак після закріплення, лише виконавець може змінювати статус завдання, змінити виконавця можна лише якщо учасник сам покинув завдання, однак для статистичного аналізу ця дія припише виконавцю одне прострочене завдання, що погіршить оцінку користувача на всій платформі.

1.1.2 Опис функціональної моделі

Для роботи розробленої системи необхідні голова проекту та учасники, що будуть виконувати завдання, результатом роботи системи в такому випадку буде рейтингові бали всіх учасників. Для цього механізмам управління в моделі будуть завдання, що поставили учасники та голова проекту, терміни виконання цих завдань, та, якщо необхідно, штрафи або винагороди. Самим механізмом моделі звичайно будуть учасники і голова

Рисунок 1.1 – Взаємодія з системою користувачів.

1.2 Огляд наявних аналогів

Серед аналогів можна виділити систему відслідковування помилок Jira, окремий функціонал якої дозволяє використовувати її як систему контролю

проектів, однак через те, що Jira не була задумана як система контролю проектів, її використання призводить до ряду незручностей, таких як окрема реєстрація як на сервісі так і на проектах, відсутній голова проекту, що може переглянути статистку проекту і вирішувати відповідно неї, які дії слід прийняти, і, саме головне, Jira це платний продукт.

Також відомим представник систем управління проектами є YouGile, однак він також є платним продуктом і сфокусований на виконані проектів через Agile методологією.

Табл.1.1 – Порівняння існуючих аналогів

Назва системи	Безкоштовність	Прив'яко до конкретної методології	Рейтингова система оціювання	Зручний у використанні
Jira	-	-	-	+
YouGile	-	+	-	+
Project Management	+	-	+	-

1.3 Постановка задачі

1.3.1 Призначення розробки

Інформаційна система технології менеджменту проектів для спрощення контролю виконання поставлених на проекті задач та завдань, а також для покращення звітності виконаної роботи

1.3.2 Цілі та задачі розробки

Для отримання необхідного результату від системи що розробляється слід максимізувати її показники ефективності. Для цього слід визначитись з взаємодією користувачів з PM. Звичайно, найкращий багато-поточний

застосунок, що здатен обробляти запити багатьох користувачів одночасно – це веб-застосунок.

Основним показник ефективності розробленої інформаційної системи є підвищення рівня контролю за виконанням та управлінням проектами. Ефективність підприємств, що використовуватимуть дану систему контролю та управління проектами однозначно збільшиться, адже голова проектів буде мати розширену звітність щодо виконаної роботи.

Висновок до розділу

В даному розділі було описано предметну область розробленої інформаційної системи її та проблеми які вона може вирішити. Були приведені системи – аналоги. Був описано процес діяльності розробленої інформаційної системи.

2 Інформаційне забезпечення

2.1 Вхідні дані

На вхід розроблена система приймає проект з його назвою та описом, користувач з його даними про себе, завдання з дататми як створення так і прострочки та складіністтю виконання та опис ролей.

Розроблена система отримує всі ці дані завдяки заповненим користувачами формам на фронт-енд, які пілся обробки відправляються на бек-енд.

2.2 Вихідні дані

Вихідними даними розробленої системи є оцінки користувачів, які система виставила базуючись на кількості виконаних вчасно завдань та їх складності і завдань від яких користувач відмовився, також розроблена система динамічно змінює статус як завданяь так і проектів згідно дій користувача, і вже згідно статусу як проекту так і завдання надавати нові необхідні дії користувачу.

2.3 Структура масивів інфомації

Розроблена система включає в себе роботу фронт-енду та бек-енду, однак для коректної роботи, спілкування між цими обома частинами розробленої системи повинно бути максимально налагоджене.

Кожен запит фронту як і відповідь серверу передають якусь модель у форматі JSON об’єкту, звичайно структурні моделі як на фронті так і на беці повинні бути для кожного з видів запитів. На беці це моделі сутностей БД які потім мапують для відправки на фронт і з точно такими же полями існують моделі на фронті, це се необхідно для кастування JSON об’єктів до нормальних типів з якими можна працювати а не парисити щоразу JSON.

2.4 Опис структури бази даних

Для зберігання даних було обрано .mdf файл. Нижче на Рис. 2.1 представлена схема бази даних розробленої інформаційної системи.

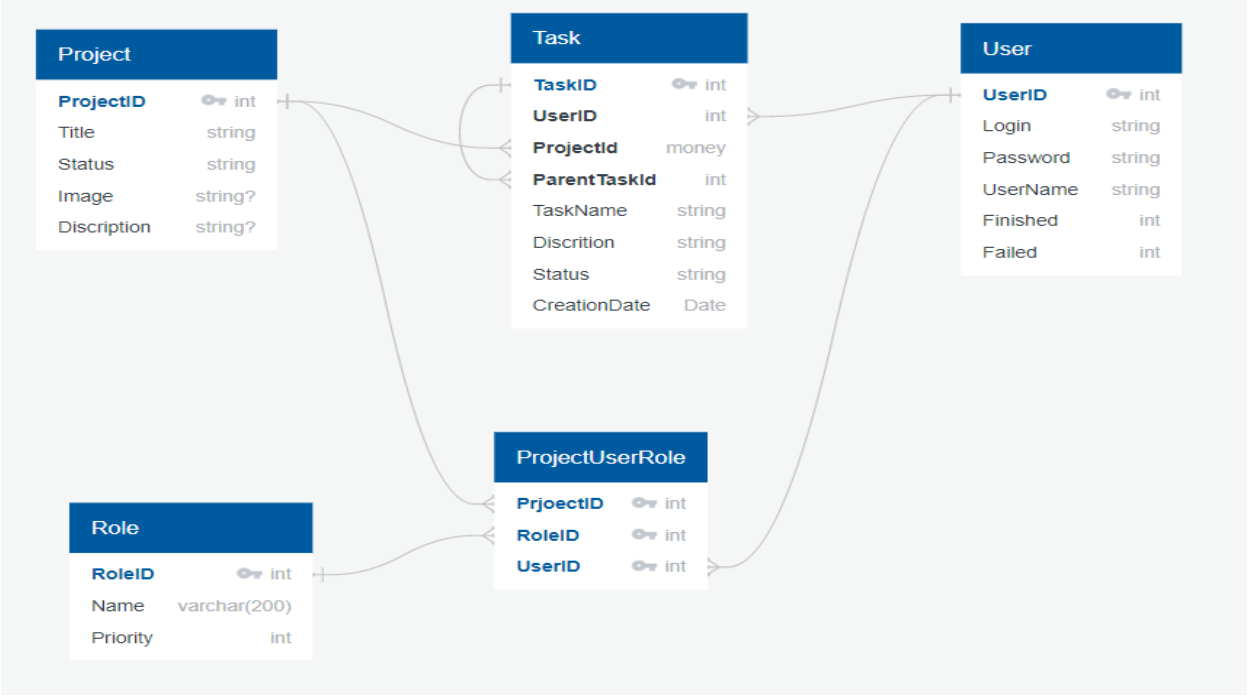


Рисунок 2.1 Схема бази даних

Нижче у таблиці 2.1 наведено сутності з яких складається база.

Таблиця 2.1 – Сутності бази даних

№	Назва таблиці	Назва сутності
1	Projects	Проекти
2	ProjectsUserRole	Зв’зок проектів з користувачами та їх ролі на проекті
3	Roles	Ролі користувачів
4	Tasks	Завдання на проекті
5	Users	Користувачі

В таблиці 2.2 детально описано представлені сутності у вигляді таблиць та параметри таблиць.

Таблиця 2.2 – Опис таблиць бази даних

Назва таблиці	Назва параметру	Тип даних	Опис

Projects	ProjectID	int	Ідентифікатор проекту
	Title	Varchar(50)	Назва проекту
	Status	Varchar(50)	Статус проекту
	Image	Varchar(100)	Презентаційний рисунок проекту.
	Discription	Varchar(100)	Презентаційний опис проекту.
ProjectsUserRole	Id	int	Ідентифікатор сутності
	ProjectId	int	Ідентифікатор проекту
	RoleId	int	Ідентифікатор ролі
	UserId	int	Ідентифікатор користувача
Roles	RoleId	int	Ідентифікатор ролі
	RoleName	Varchar(50)	Назва ролі
	Priority	int	Пріоритет ролі
Tasks	TaskId	int	Ідентифікатор завдання
	UserId	int	Ідентифікатор користувача
	ProjectId	int	Ідентифікатор проекту
	ParentTaskId	int	Ідентифікатор батьківського завдання
	TaskName	Varchar(50)	Назва завдання
	Discription	Varchar(100)	Опис завдання

	Status	Varchar(50)	Статус завдання
	CreatDate	DateTime	Дата та час створення завдання
	EstimateDate	DateTime	Дата та час коли завдання стане простроченим
Users	UserId	int	Ідентифікатор користувача
	Rating	float	Рейтинг користувача
	Login	Varchar(50)	Логін користувача
	Password	Varchar(50)	Пароль користувача
	UserName	Varchar(50)	Нік-нейм користувача
	Finished	int	К-сть завершених завдань
	Failed	int	К-сть завдань від яких відмовився користувач
	Expired	int	К-сть прострочених завдань

Висновок до розділу

В даному розділі було описано вхідні та вихідні параметри рохробленої системи. Наведено структуру бази даних та характеристику кожної таблиці БД.

3 Математичне забезпечення

3.1 Змістовна постановка задачі

Для менеджерів проекту, щоб вирішити чи слід закріпляти дану задачу за конкретним користувачам або чи слід змінити його роль на проекті і як саме, з використанням розробленої системи не потрібно мати дані про продуктивність робітників проекту.

Показник продуктивності необхідний для звітування, однак завдяки системі технології менеджменту, учасникам за виконані завдання нараховують рейтинг, но основі якого сама система дозволяє присвоїти завдання конкретної складності тим користувачам чий рейтинг достатньо великий. Розробляєма інформаційна система технології менеджменту автоматично виставляє ці оцінки опираючись на кількість виконаних вчасно завдань та кількість прострочених завдань.

Слід зауважити, що якщо учасник відмовився від завдання його рейтинг знижується на величину незалежну від складності завдання, однак залежну від кількості рейтингових очок учасника.

3.2 Математична постановка задачі

Нехай на у проекті приймають участь n учасників, кожен з яких виконав по m завдань однакової складності вчасно, після чого відмовились від h завдань. Задача полягає в тому, щоб змінити рейтинг всіх учасників згідно з виконаними та відмовленими завданнями та вирішити які завдання, що не були виконані на проекті можуть присвоїти собі всі n учасників, якщо їх початковий рейтинг дорівнює нулю.

3.3 Обґрунтування методу розв’язання

Спосіб оцінювання по кривій досвіду не надто комплексний однак найбільш наглядний, тому він і використовують в більшості систем прогресу

в різним інформаційних системах. Тому було вирішено запровадити саме змінну оцінку, щоб для користувачів з високим рейтингом прості завдання давали менше користі ніж для користувачів з низьким.. З альтернативних варіантів вирішення поставленої задачі можна було додати коефіцієнт ролі користувача, що на низьких рівнях покращувало би оцінку а на високих рівнях ні, однак користувачі можуть бути учасниками різних проектів на різних ролях, що робить такий метод дуже складним для реалізації.

3.4 Опис методів розв'язання

Нехай в системі присутній 1 проект з 2ома учасниками які вже встигли виконати по 2 завдання однаковим початковим рівнем складності, виконання яких дарувало учасникам по 5 одиниць рейтингу.

Оскільки рейтинг в програмі обчислюється по кривій досвіду.

Формула отриманих очок рейтингу за виконане завдання має наступний вигляд:

E += X / (E / 2) , E > 1 .

E += X, E = 0 .

Де E – к-сть очок рейтингу користувача,

X – к-сть очок рейтингу за вчасно виконане завдання.

Вчасно 2 завдання виконав лише перший учасник, тому його рейтинг = 7 одиниць, а другий лише одне завдання виконав вчасно, тому його рейтинг = 5 одиниць. На проекті присутні два завдання: одне з початковим рівнем складності(необхідно 0 та більше очок рейтингу), та з середнім рівнем складності(необхідно 6 та більше очок рейтингу). Системи, що порахувала рейтинг обох користувачів, надає змогу користувачу обрати обидва завдання, однак чим складніше завдання він візьме тим більше очок рейтингу отримає і разом з цим змогу брати більш складні завдання.

Висновок до розділу

У розділі математичного забезпечення були оглянуті математичні задачі дипломного проекту. Оглянуті різні варіанти вирішення поставлених задач. Обгрутовано вибір поточних рішень.

4 Програмне та технічне забезпечення

4.1 Засоби розробки

Для виконання поставлених задач проект було вирішено розділити на два під-проекти: бек-енд та фронт-енд.

Для розробки бек-енд проекту найбільш функціональним та гнучкою специфікацією є Web Api, що дозволяє отримувати та відправляти запити в форматі JSON, що дозволить наладити спілкування клієнту та серверу доволі просто. Для розробка бек-енд проекту було вирішено використовувати мову програмування C#, адже він має всі необхідні та достатні функціонал для вирішення поставлених задач в розробці бек-енд проекту.

Для розробки фронт-енд проекту найбільш функціональною є мова програмування JavaScript, але використання Angular Framework, що є відкритою бібліотекою для JavaScript, вимушує використовувати TypeScript – який є більш розширеним ніж JavaScript для розробки веб-застосунків. Для вирішення поставлених задач був використаний Angular 9, що був доповнений новими можливостями для формування клієнтських запитів.

4.2 Вимоги до технічного забезпечення

Мінімальні вимоги для технічного забезпечення клієнту:

RAM: 512Мб,

Місце на диску: 200Мб,

Процесор: одно-ядерний, 1,2ГГц,

Відеокарта: 512Мб,

Монітор: будь-який,

Браузер: Google Chrome

Рекомендовані вимоги для технічного забезпечення клієнту:

RAM: 1024Мб,

Місце на диску: 200Мб,

Процесор: одно-ядерний, 1,5ГГц,

Відеокарта: 512Мб,

Монітор: будь-який,

Браузер: Google Chrome

Мінімальні вимоги для технічного забезпечення серверу:

RAM: 1024Мб,

Місце на диску: 1200Мб,

Процесор: одно-ядерний, 1,5ГГц

Рекомендовані вимоги для технічного забезпечення серверу:

RAM: 1024Мб,

Місце на диску: 1200Мб,

Процесор: одно-ядерний, 1,5ГГц

4.2.1 Загальні вимоги

Вміння користуватись браузером Google Chrome,

Розуміння англійської мови.

4.3 Архітектура програмного забезпечення

4.3.1 Діаграма класів

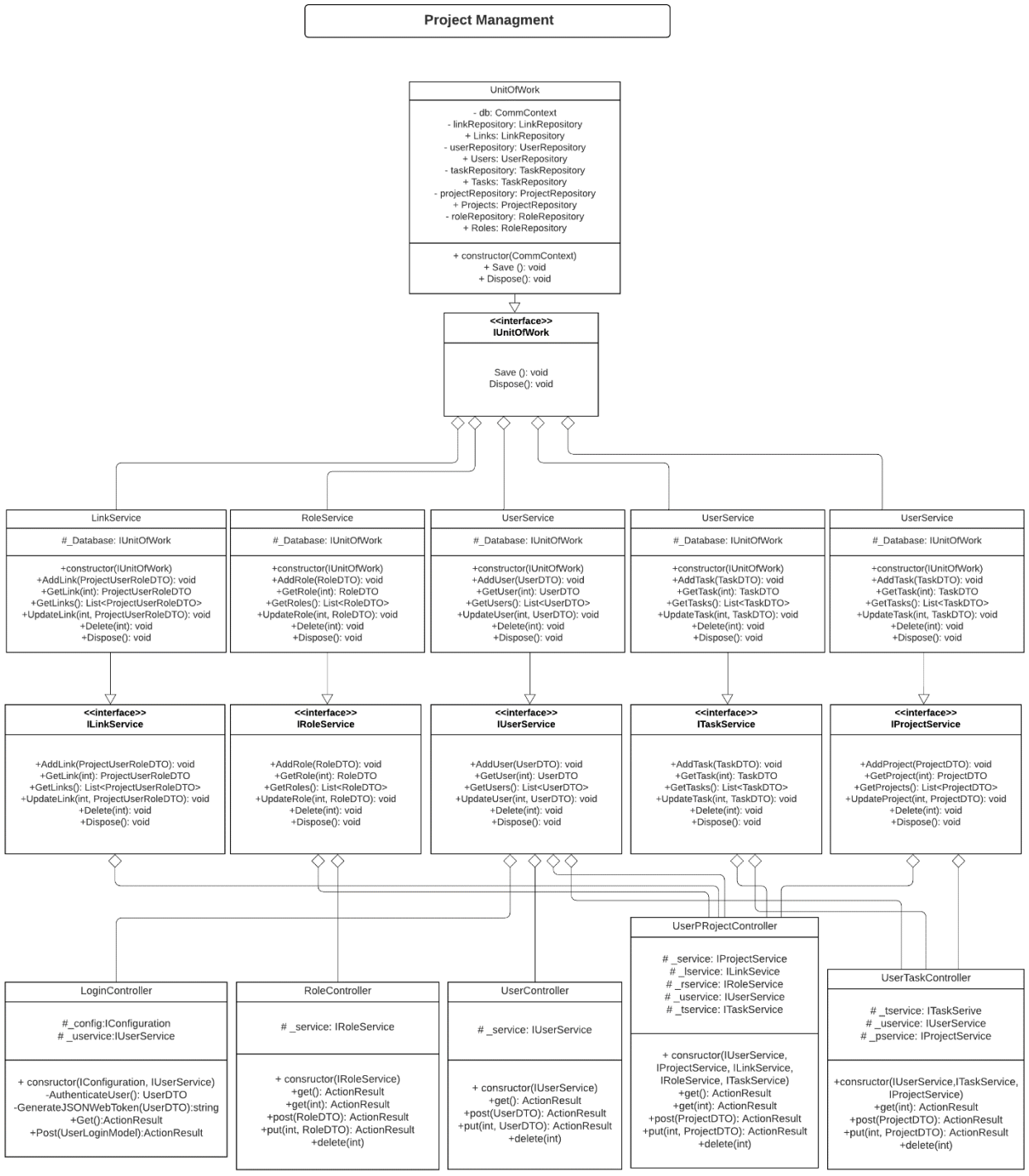


Рисунок 4.1 - Діаграма класів одинці роботи бек-енд проекту

На даній діаграмі класів зображена, так звана одиниця роботи бек-енд проекту, що робить якусь одну з прописаних дій з потрібною моделлю, що

прийшла з фронт-енду, після чого звільнює пам’ять, що значно полегшує роботу серверу. Також на діаграмі можна побачити, що всі зв’язки реалізовані через інтерфейси та ін’єкції залежностей, що робіть цей проект максимально слабко зв’язним і дає змогу багатьох реалізацій всіх інтрефейсів без шкоди для проекту. Звичайно в проекті присутні класи контексту бази даних та класи моделі, що відповідають сутностям БД. Для класу контексту бази даних були розроблені класи репозиторії які надають змогу працювати лише з 1ю сутністю за 1ну одиницю роботи, це покращує безпечність коду.

Таблиця 4.1 – Опис додаткових класів бек-енд проекту

Назва додаткового класу	**Опис**
CommContext	Клас контексту бази даних, саме в ньому зберігається вся інформація з БД, та поповнюється за допомогою класів репозиторіїв
Project	Клас модель проекту для зв’язку з БД
ProjectDTO	Клас модель проекту для зв’язку з фронт-ендом
ProjectRepository	Клас репозиторій сутності проектів, відповідає за всі дії з сутністю проектів у БД
User	Клас модель користувача для зв’язку з БД
UserDTO	Клас модель користувача для зв’язку з фронт-ендом
UserRepository	Клас репозиторій сутності користувачів, відповідає за всі дії з сутністю користувачів у БД
ProjectUserRole	Клас модель зв’язку проектів, користувачів, ролей проекту для зв’язку з БД
ProjectUserRoleDTO	Клас модель зв’язку проектів, користувачів, ролей проекту для зв’язку з фронт-ендом
LinkRepository	Клас репозиторій сутності зв’язку проектів, користувачів, ролей проекту, відповідає за всі дії з цією сутністю у БД

Task	Клас модель завдання для зв’язку з БД
TaskDTO	Клас модель завдання для зв’язку з фронт-ендом
TaskRepository	Клас репозиторій сутності завдань, відповідає за всі дії з сутністю завдань у БД
Role	Клас модель ролі для зв’язку з БД
RoleDTO	Клас модель ролі для зв’язку з фронт-ендом
RoleRepository	Клас репозиторій сутності ролей, відповідає за всі дії з сутністю ролей у БД

На фронт-енд проекті не існує прямої залежності між класами, адже вони не мають своїх екземплярів які визивавали б інші компоненти, тому в таблиці 4.2 представлений список класів фронт-енд проекту та їх опис.

Таблиця 4.2 - Опис додаткових класів фронт-енд проекту

Назва додаткового класу	**Опис**
Main	Головний файл проекту, з нього починається виконання
AppComponent	Клас компонента всього застосунку, в ньому знаходяться всі інші компоненти
LoginComponent	Компонента для заповнення логін моделі користувача
RegisterComponent	Компонента для заповнення реєстраційної моделі користувача
UserProjectComponent	Компонента, що відображає всі проекти на які підписаний користувач та дає змогу їх редагувати або покидати
UserTaskComponent	Компонента, що відображає всі завдання на які підписаний користувач або які існують на проекті та дає змогу їх редагувати або покидати
ProfileComponent	Компонента що дає змогу вийти з застосунку та передивитись особисті дані такі як рейтинг
RoleComponent	Компонента, що відповідає за відображення та редагування ролей.

UserComponent	Компонента, що відповідає за відображення та редагування користувачів.
LoginUser	Клас модель форми логіну користувача
Project	Клас модель форми проекту
ProjectUserRole	Клас модель форми зв’язку між моделями проекту користувачу та ролі
Puttask	Клас модель форми зміни моделі завдання
RegiseterUser	Клас модель реєстраційної форми користувача
Role	Клас модель ролі користувача
Task	Клас модель завдання
User	Клас модель користувача

4.3.2 Діаграма послідовності

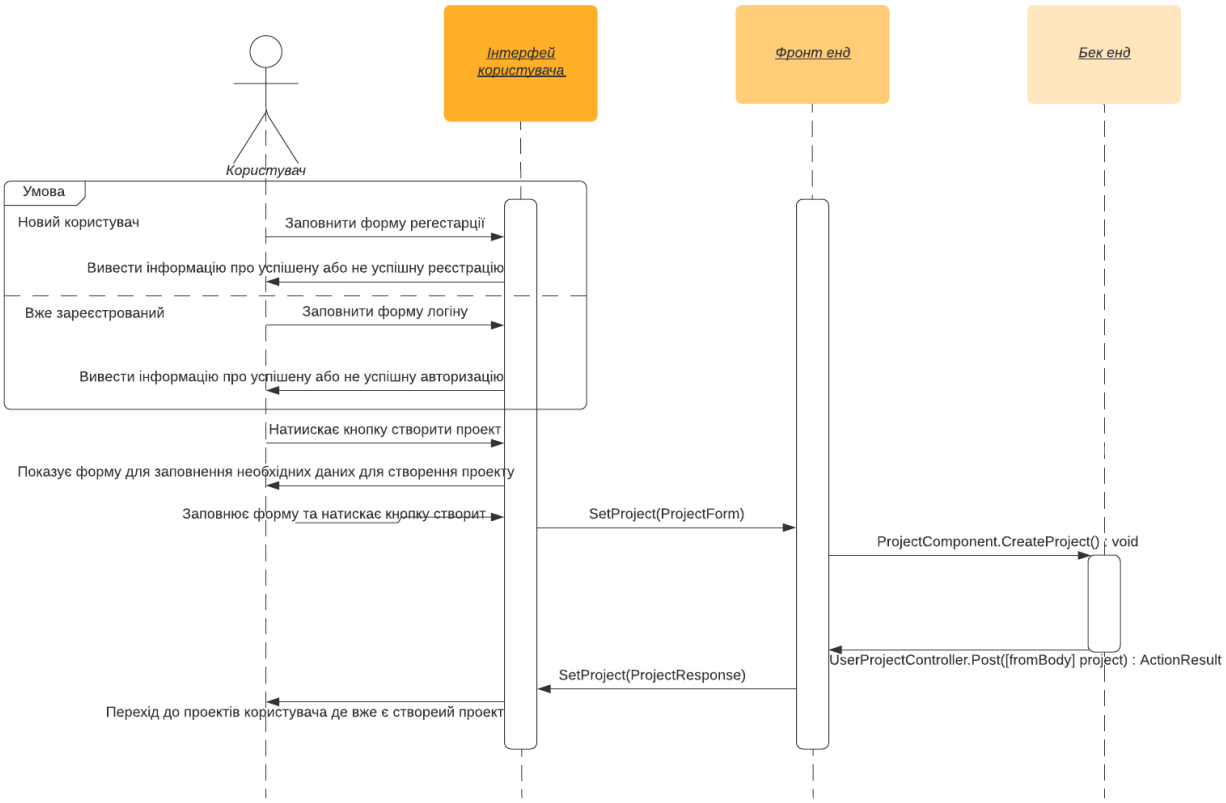


Рисунок 4.2 – Діаграма послідовності створення нового проекту в системі.

На Рис. 4.2 зображено процес створення нового проекту у системі, що передбачає собою створення фронт-ендом та заповнення користувачем

форми яка вмістить всю необхідну інформацію про новий проект. Заповнивши форму створення нового проекту, користувач віддає проект фронт-енду, який за допомогою функції SetProject устанавлює полю проекту значення які ввів користувач, після чого фронт-енд визиває функцію CreateProject, що відправляє запит на сервер. В запиті до серверу буде токен користувача що дозволить ідентифікувати його та закріпити новий проект саме за ним, та надати статус боса. Після додання проекту в БД в методі POST фронт-енду повертається AtionResult з новим вже створеним проектом, який фронт устанавлює полем проекту і демонструє користувачу.

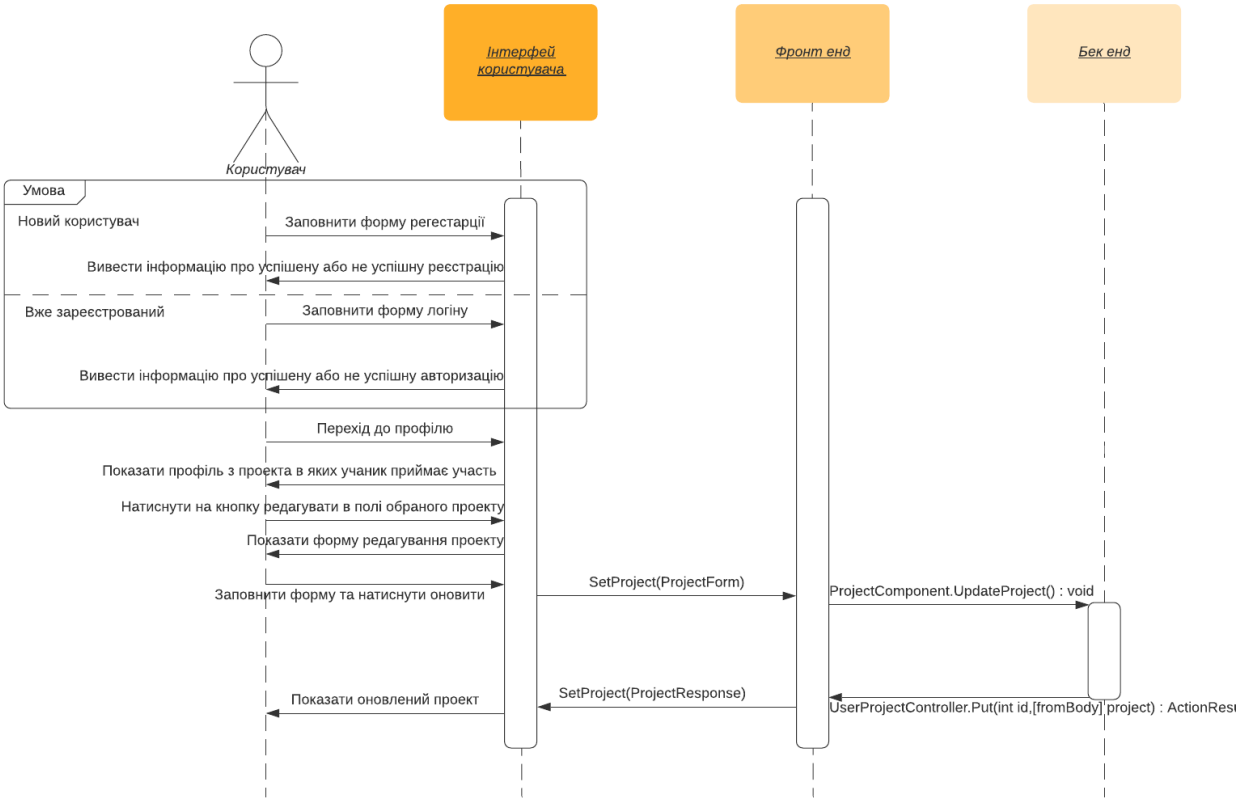


Рисунок 4.3 – Діаграма послідовності редагування вже існуючого проекту.

На Рис. 4.3 видно, що для редагування існуючого проекту необхідно ввійти в свій профіль де буде перераховані та представлені всі проекти в яких учасник приймає участь та представлена можливість на редагування. Після натиснення на кнопку редагування проекту користувач заповнює форму з новими даними проекту і зберігає їх, таким чином на проекті установлюється змінений проект за допомогою функції SetProject, після чого визивається

функція UpdateProject, яка формує запит для зміни даних проекту та відправляє його на сервер, той приймає запит і функцією PUT відповідає про успішне чи не успішне оновлення проекту, після чого встановлюється вже оновлений проект на фронт-енді та демонструється користувачу.

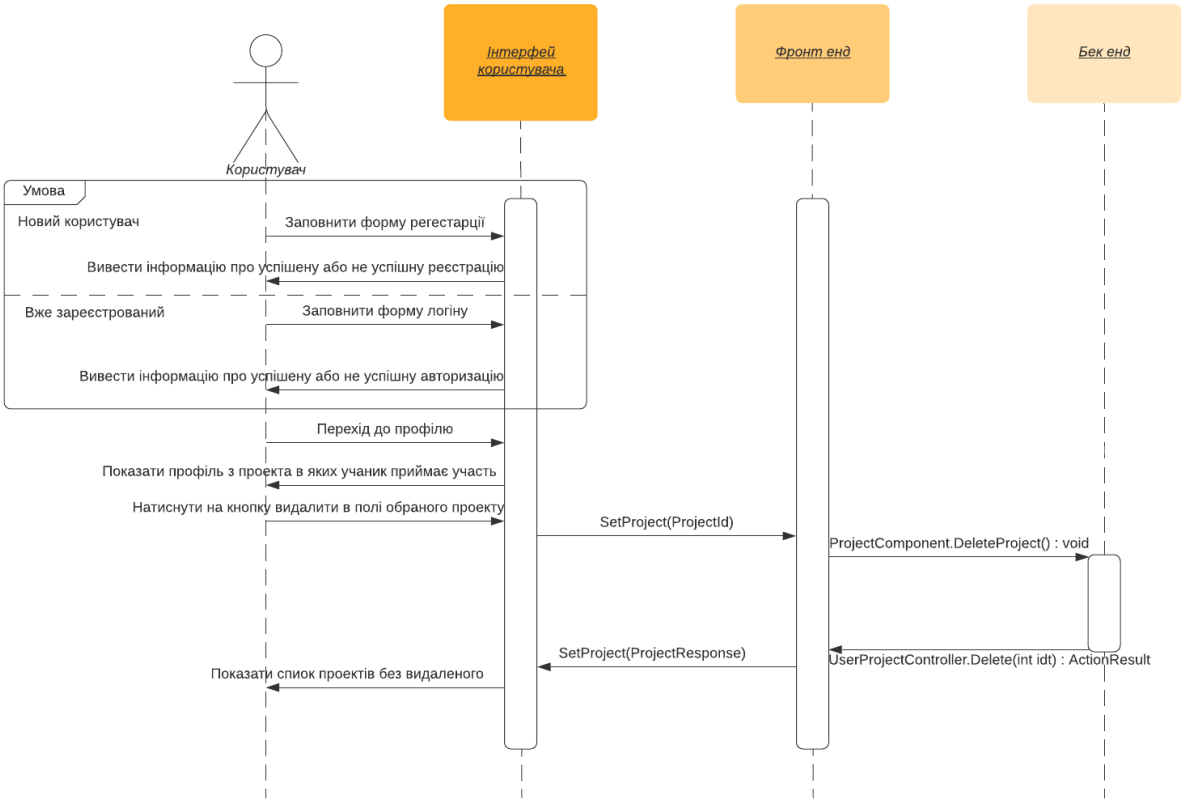


Рисунок 4.4 – Діаграма послідовності видалення проекту з системи

На Рис. 4.4 видно, що як і для редагування слід зайти в свій профіль де є всі ваші проекти та натиснути замість кнопки редагувати кнопку видалити, що видалить проект з БД а разом з ним всі зв’язки користувачів та ролей на цьому проекті.

Натиснув кнопку видалити проект, користувач не заповню форм функція SetProject отримує незмінений проект з компоненти та установлює його своїм полем, пілся чого формує запит, який бек-енд отримує та обробляє функцією DELETE після чого надсилаю булеву зміну про удачність виконання операції. Якщо проект було видалено функція SetProject виставляє полю проект значення NULL, що не дозволить виводити ніякої інформації про видалений проект.

Звичайно кейси з іншими сутностями передається на основі проектів адже всі сутності так чи інакше залежать від проекту.

4.3.3 Діаграма компонентів

Рисунок 4.5 – Діаграма компонентів системи технологій менеджменту проектів.

На Рис. 4.5 зображені основні компоненти системи технологій менеджменту проектів, а саме: Фронет-енд, що містить Angualr який звертається до контролерів бек-енду та інтерфейс користувача, який складається з таблиць стилів CSS та HTML; Наступна компонента системи це бек-енд, який спілкується з фронт-ендом завдяки контролерам, також має моделі сутностей з БД та сервісу по взаємодії з цими моделями; і нарешті компонента бази даних яка містить лише файл БД як сховище всіх даних, який потім можна захостити на великому SQL сервер.

4.3.4 Специфікація функцій

Таблиця 4.3 – Специфікація функцій бек-енду.

Клас	**Назва функції**	**Опис дії**
LoginController	public LoginController(IUserService service, IConfiguration conf)	Конструктор що задає перелік сервісів які буде використовувати контролер для зберігання або редагування даних БД.
	private UserDTO AuthenticateUser(UserLoginModel loginModel)	Перевіряє вірність паролю при авторизації користувача у систему.
	private string GenerateJSONWebToken(UserDTO userInfo)	Генерує веб-токен, що буде відправлятись в хедерах запиту до бек-енду, і який має всю необхідну інформацію для ідентифікації користувача, зберігання такого токену в Local Storage браузеру дозволяє не авторизуватись щоразу як покинули сторінку з системою.
	public ActionResult Get()	Повертає перелік користувачів, що були зареєстровані в системі у виді списку.
	public ActionResult Post([FromBody]UserLoginModel login)	Викликає метод AuthenticateUser і якщо пароль вірний викликає GenerateJSONWebToken, після чого повертає токен на фронт-енд де він вже

		зберігається на Local Storage браузеру.
RoleController	public RoleController(IRoleService service)	Конструктор що задає перелік сервісів які буде використовувати контролер для зберігання або редагування даних БД.
	public ActionResult Get()	Повертає всі ролі, що були додані до системи у виді списку.
	public ActionResult Get(int id)	Повертає конкретну роль поле ID якого співпадає с параметром який передали у функцію
	public ActionResult Put(int id, [FromBody] RoleDTO role)	Отримує змінену модель ролі з фронту та знаходить по полю ID вже існуючу в БД після чого заміняє стару на нову.
	public ActionResult Post([FromBody] RoleDTO role)	Отримує нову модуль ролі з фронту та додає її у БД.
	public ActionResult Delete(int id)	Знаходить вже існуючу модель ролі по параметру ID, що передається в параметрах функції та видаляє її.
UserController	public UserController(IUserService service)	Конструктор що задає перелік сервісів які буде використовувати контролер для зберігання або редагування даних БД
	public ActionResult Get()	Отримує з токену ID користувача, що зробив запит та повертає всю

		інформацію про користувача
	public ActionResult Put([FromBody] UserDTO user)	Отримує змінену модель користувача з фронту та знаходить по полю ID вже існуючу в БД після чого заміняє стару на нову.
	public ActionResult Post([FromBody] UserDTO user)	Отримує нову модель користувача з фронту та додає її у БД.
	public ActionResult Delete(int id)	Знаходить вже існуючу модель користувача по параметру ID, що передається в параметрах функції та видаляє її.
UserProjectControl ler	public UsersProjectController(ITaskSer vice tservice, IRoleService rservice, IProjectService service, ILinkService lservice, IUserService uservice)	Конструктор що задає перелік сервісів які буде використовувати контролер для зберігання або редагування даних БД
	public ActionResult Get()	Повертає всі проекти в яких приймає участь зареєстрований учасник системи в виді списку.
	public ActionResult Get(int id)	Повертає або всіх учасників конкретного проекту або всі завдання, що були створені на цьому проекті у виді списку.
	public ActionResult Post([FromBody]ProjectDTO project)	Отримує нову модель проекту з фронту та додає її у БД.
	public ActionResult Put(int id,[FromBody]ProjectDTO project)	Отримує змінену модель проекту з фронту та знаходить по полю ID вже

		існуючу в БД після чого заміняє стару на нову або звільнює користувача з проекту або змінює виконавця завдання, що було створена на проекті.
	public ActionResult Delete(int id)	Знаходить вже існуючу модель проекту по параметру ID, що передається в параметрах функції та видаляє її також видаляє всі моделі зв’язків проекту користувача та ролі.
UserTaskController	public UserTaskController(ITaskService tservice, IUserService uservise, IProjectService pservice)	Конструктор що задає перелік сервісів які буде використовувати контролер для зберігання або редагування даних БД
	public ActionResult Get(int Id)	Повертає список всіх завдань на проекті за якими закріплений учасник системи, що відправив запит.
	public ActionResult Post([FromBody] TaskDTO task)	Створює на проекті нове завдання без закріпленого користувача.
	public ActionResult Put(int id, [FromBody] TaskDTO task)	Отримує змінену модель завдання з фронту та знаходить по полю ID вже існуючу в БД після чого заміняє стару на нову.
	public ActionResult Delete(int id)	Знаходить вже існуючу модель завдання по параметру ID, що передається в

		параметрах функції та видаляє її.

Таблиця 4.4 – специфікація функцій фронт-енду

Клас	**Назва функції**	**Опис дії**
AppComponent	ngOnInit(): void	Відправляє запит на сервер якщо Local Storage зберіг токен та установлює користувача з відповіді сервера.
	LogedIn(vr: boolean): void	Відправляє запит на всі проекти користувача та встановлює їх з відповіді серверу.
LoginComponent	async Save()	Відправляє запит на авторизацію користувача.
ProfileComponent	async Delete()	Відправляє запит на видалення користувача з системи.
	async Save()	Відправляє запит на зміну даних користувача.
RegisterComponent	async Save()	Відправляє запит на створення нового користувача в системі.
RoleComponent	DeleteProject(): void	Відправляє запит на видалення ролі з системи.
	Save(): void	Відправляє запит на зміну даних ролі.
UserComponent	DeleteProject(): void	Відправляє запит на видалення користувача з системи.
	Save(): void	Відправляє запит на зміну даних користувача.
ProjectComponent	Delete(id: number): void	Відправляє запит на видалення проекту з системи.
	Save(i:number): void	Відправляє запит на зміну даних проекту.

	Leave(projId: number):void	Відправляю запит на видалення зв’язку користувача з проектом.
	ManageUsers():void	Відправляє запит на отримання всіх користувачів на проекті та встановлює відповідь у виді списку користувачів.
	ManageTasks():void	Відправляє запит на отримання всіх завдань на проекті та встановлює відповідь у виді списку завдань.
	FireUser():void	Відправляю запит на видалення зв’язку користувача з проектом.
	ChangeUserOnTask(): void	Відправляє запит на зміну поля UserId в моделі завдання.
UserTaskComponent	onChange(v: string): void	Відправляє запит на отримання всіх завдань користувача на проекті та встановлює відповідь у виді списку.

Висновок до розділу

Розробляючи цей розділ було побудована діаграма класів, послідовності та компонентів, що представило інформаційну систему з точки зору проектування. Була описана специфікація функцій класів, як фронт енду так і бек енду системи, яка дозволяє простіше зрозуміти функціональну структуру класів.

5 Технологічний розділ

5.1 Керівництво користувача

На початку роботи користувача з системою йому необхідно зареєструватись у системі. Відкривши застосунок користувач побачить, форму авторизації зображеної на Рис.5.1 однак на ній існує можливість відкрити форму реєстрації(див. Рис5.2). Логін користувача повинен бути унікальним.

Рисунок 5.1 – Форма авторизації

Рисунок 5.2 – Форма реєстрації

Після реєстрації даних нового користувача, необхідно виконати авторизацію, після чого обробивши запит сервер відповість чи вірні дані були введені.

Після авторизації перед користувачем відкриється профіль з його даними та проектами в яких він приймає участь, у проектах буде показан роль користовача та буде доступний відповідний його ролі функціонал як зображено на Рис.5.3

Рисунок 5.3 – Профіль користувача з його даними та проектами в яких той приймає участь.

Якщо користувач захоче створити новий проект він повинен натиснути на кнопку Projects, що замынила собою кнопку Login в хедері проекту(див. Рис5.4)

Рисунок 5.4 – Хедер після авторизації користувача.

Як тільки користувач натисне кнопку Projects він побачить всі прокти системи в яких він не приймає участі(див Рис.5.5) та кнопку, що створює новий проект.

Рисунок 5.5 – Перелік проектів в яких користувач не приймає участі.

Якщо, користувач зацікавився проектом то ві може прийняти в ньому участь (після натиснення кнопки Join проект з’явиться у профілі (дви. Рис.5.6)), однак спочатку його роль на проекті завжди дорінює 0(Worker), для її зміни необхідний дозвіл голови проекту.

Рисунок 5.6 – Проект в якому зацікавився учасник доданий до переліку.

Якщо учасник захотів змінити свої дані він може натиснути кнопку Update в своєму профілі та змінити дані(див Рис5.7)

Рисунок 5.7 – Форма зміни даних з нови UserName користувача з логііном User1.

Якщо натиснути кнопку Save, тоді зміни збережутся для цього користувача (див. Рис5.8).

Рисунок 5.8 – Нові дані користувача User1 в його профілі

Якщо натиснути кнопку Loggoff ви вийдите з свого профілю і буде необхідно знову проводити авторизацію.

Якщо ви босс проекту то вам доступно більше можливостей(див Рис.5.6). User1 є босом 1го проекту зі списку його проектів тому він може

змінити дані проекту, для цього необхідно натиснути кнопку Update Project після чого зможе міняти дані проекту(див. Рис5.9).

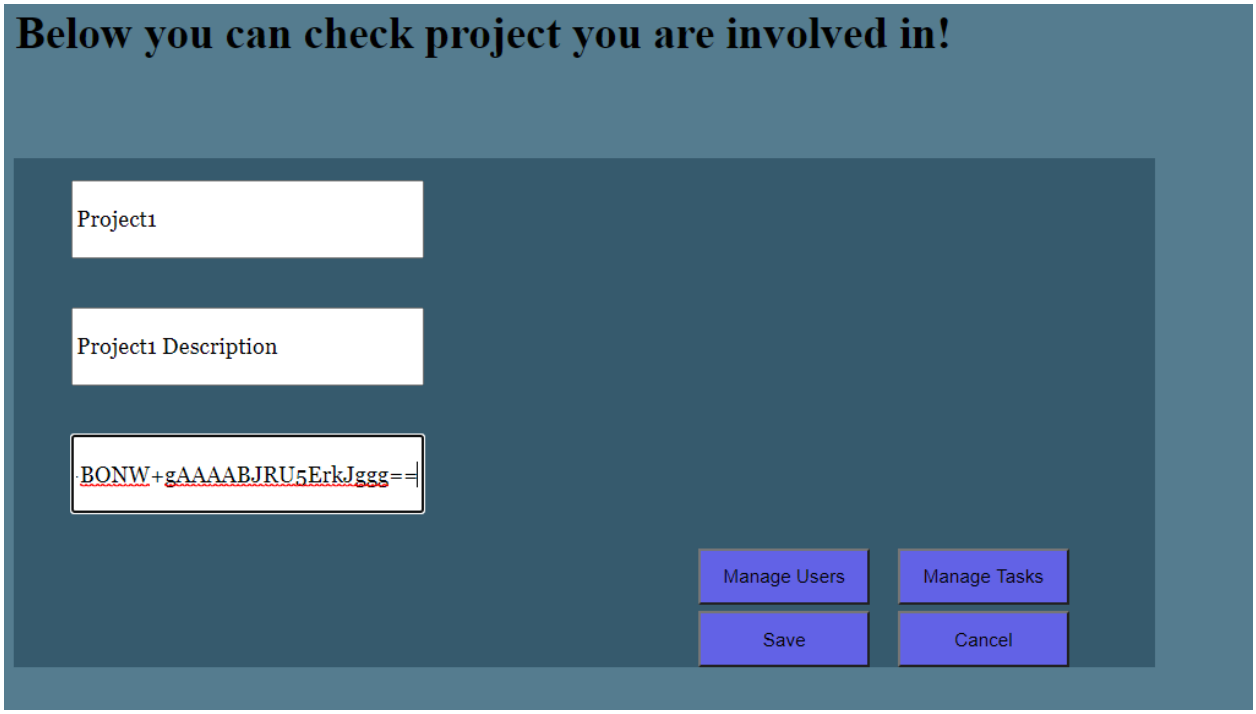


Рисунок 5.9 – Форма даних з доданою картинкою до проекту.

Як і в профілі якщо, натиснути кнопку Save тоді дані збережуться(див Рис.5.10).

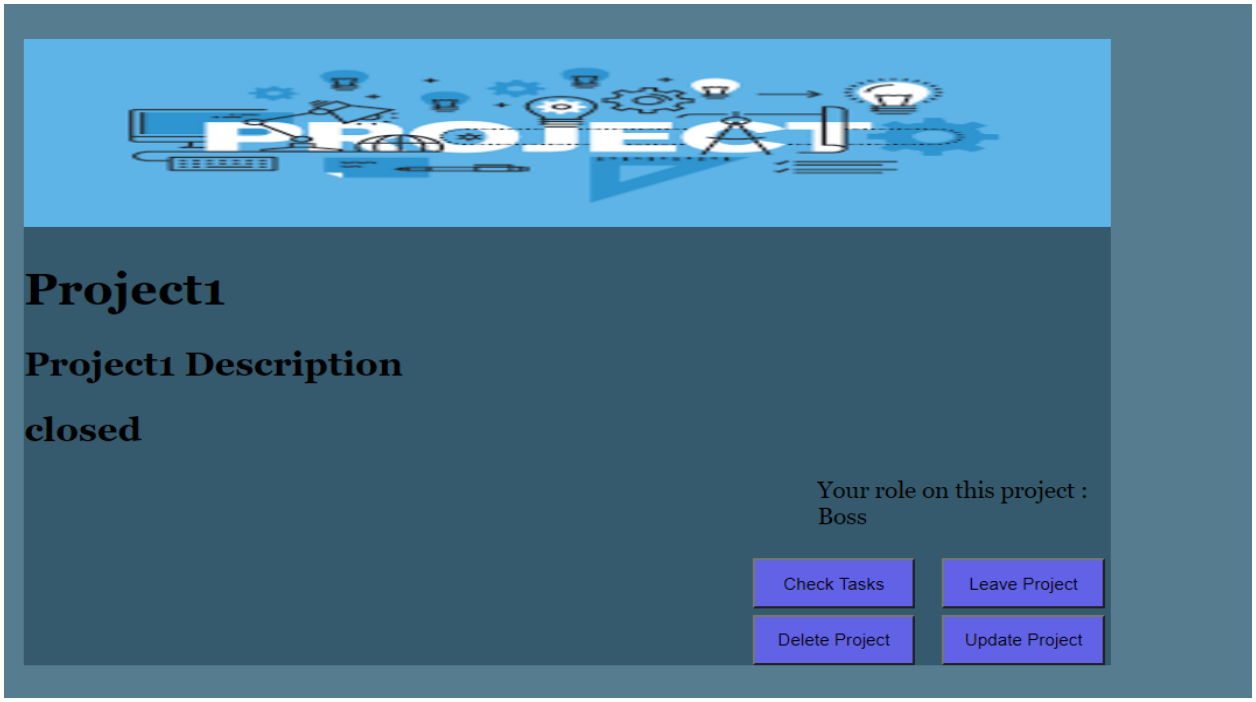


Рисунок 5.10 – Той самий проект з збереженою картинкою, щоб зацікавити інших користувачів.

Користувач може передивитись(див. Рис.5.11) та змінити статус або відмовитись від завдань на проекті(див. Рис.5.12).

Рисунок 5.11 – Селектор тасків які хоче передивитись користувач.

Рисунок 5.12 – Завдання User1 на проекті 1.

Як видно на Рис.5.12 1 з завдань, що даэ 5 очок рейтингу вже завершено, тому у профілі користувача(див. Рис.5.3) рейтинг дорівнює п’яти.

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Метою випробувань являється перевірка відповідності функцій комплексу задач інформаційної системи технологій менеджменту проектів вимогам технічного завдання.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем;
– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів.

5.2.3 Результати випробувань

В процесі тестування була перевірена уся функціональність комплексу задач (КЗ). У наступних таблицях наведений перелік випробувань основних функціональних можливостей та результати, що свідчать про відповідність чи невідповідність функцій вимогам.

Таблиця 5.1 – Результати перевірки функції авторизації

Мета тесту	Перевірка функції авторизації
Початковий стан КЗ	Відкрите вікно авторизації
Вхідні данні	Логін та пароль користувача
Схема проведення тесту	Ввести логін і пароль у відповідні поля
Очікуваний результат	Відкритий профіль користувача та його проекти
Стан КЗ після проведення випробувань	Відкритий профіль користувача та його проекти

Таблиця 5.2 – Результати перевірки функції реєстрації

Мета тесту	Перевірка функції реєстрації
Початковий стан КЗ	Відкрите вікно реєстрації
Вхідні данні	Логін, пароль та юзер нейм користувача

Схема проведення тесту	Ввести логін і пароль і юзер нейм у відповідні поля
Очікуваний результат	Відкритий форма авторизації
Стан КЗ після проведення випробувань	Відкритий форма авторизації

Таблиця 5.3 – Результати перевірки функції зміни даних профілю

Мета тесту	Перевірка зміни даних профілю
Початковий стан КЗ	Відкрите вікно профілю
Вхідні данні	Логіна або пароль або юзер нейм користувача
Схема проведення тесту	Ввести логін або пароль або юзер нейм у відповідні поля
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.4 – Результати перевірки функції зміни даних проекту

Мета тесту	Перевірка зміни даних проекту
Початковий стан КЗ	Відкрите вікно профілю
Вхідні данні	Назва проекту або опис проекту або картинка для проекту
Схема проведення тесту	Ввести назву проекту або опис проекту або картинку для проекту у відповідні поля
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.5 – Результати перевірки функції зміни даних завдання

Мета тесту	Перевірка зміни даних завдання
Початковий стан КЗ	Відкрите вікно профілю
Вхідні данні	Статус завдання
Схема проведення тесту	Натиснути кнопку з протилежним значенням до теперішнього статусу завдання
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.6 – Результати перевірки функції додання проекту до списку проектів користувача

Мета тесту	Перевірка зміни наповнення списку проектів користувача
Початковий стан КЗ	Відкрите вікно проектів в яких користувач не приймає участі
Вхідні данні	Всі поля проекту
Схема проведення тесту	Натиснути кнопку Join у області обраного проекту
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.7 – Результати перевірки функції додання завдання до списку завдань проекту

Мета тесту	Перевірка зміни наповнення списку завдань проекту
Початковий стан КЗ	Відкрите вікно профілю з списком завдань на проекті
Вхідні данні	Всі поля завдання
Схема проведення тесту	Натиснути кнопку Add Task у області обраного проекту
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.8 – Результати перевірки функції присвоєння завдання користувачу

Мета тесту	Перевірка зміни наповнення списку завдань користувача
Початковий стан КЗ	Відкрите вікно профілю з списком завдань користувача
Вхідні данні	Ідентифікатор користувач в необхідному полі
Схема проведення тесту	Натиснути кнопку Manage Task у області обраного проекту в стані зміни завдань
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.9 – Результати перевірки функції покидання проекту

Мета тесту	Перевірка зміни наповнення списку проектів користувача
Початковий стан КЗ	Відкрите вікно профілю з проектами користувача
Вхідні данні	Ідентифікатор користувача
Схема проведення тесту	Натиснути кнопку Leave у області обраного проекту
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.10 – Результати перевірки функції нарахування очок рейтингу користувачу

Мета тесту	Перевірка функції нарахування очок рейтингу користувачу
Початковий стан КЗ	Відкрите вікно профілю з проектами користувача та обраними завданнями користувача
Вхідні данні	Ідентифікатор користувача
Схема проведення тесту	Натиснути кнопку Finish у області обраного завдання
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.11 – Результати перевірки функції відрахування очок рейтингу користувачу

Мета тесту	Перевірка функції відрахування очок рейтингу користувачу
Початковий стан КЗ	Відкрите вікно профілю з проектами користувача та обраними завданнями користувача
Вхідні данні	Ідентифікатор користувача
Схема проведення тесту	Натиснути кнопку Leave у області обраного завдання
Очікуваний результат	Відкритий профіль з новими даними
Стан КЗ після проведення випробувань	Відкритий профіль з новими даними

Таблиця 5.12 – Результати перевірки видалення користувача

Мета тесту	Перевірка функції видалення користувача
Початковий стан КЗ	Відкрите вікно профілю
Вхідні данні	Ідентифікатор користувача
Схема проведення тесту	Натиснути кнопку Delete у області профілю користувача
Очікуваний результат	Відкрите вікно авторизації без змоги зайти використавши дані видаленого корисувача
Стан КЗ після проведення випробувань	Відкрите вікно авторизації без змоги зайти використавши дані видаленого корисувача

Висновок до розділу

В даному розділу було розроблено та представлено керівництво користувача, в якому продемонстровані функції інформаційної системи у відповідності до постановки задачі, що була наведена в першому розділі, та були представлені рисунки відповідних представленим функціями графічних інтерфейсів та форм.

Програмне забезпечення(ПЗ) було протестоване, в результаті тестів якого і було встановлено, що ПЗ виконує усі функції, що були заявлені, правильно, що є демонструє ПЗ як роботоздатне.

ЗАГАЛЬНІ ВИСНОВКИ

В рамках дипломного проекту було описано та розроблено інформаційну систему технологій менеджменту проектів з використанням RestAPI “Project Management”. Розроблена система заходить своє застосування у будь-якій сфері діяльності людини в якій є голова та виконавець. Використання даної системи значно покращить та полегшить процес контролю виконання завдань на проекті та визначення кандидатів на виконання ще не виконаних завдань, що звільнить від необхідності обирати кандидатів самостійно.

В процесі розробки дипломного проекту був вирішений ряд задач, таких як:

- Створення інформаційного забезпечення;
- Розробка архітектури та проектування системи;
- Проектування сервісів;
- Проектування інтерфейсу користувача;
- Тестування розробленого продукту;

В процесі проектування системи були побудовані діаграма класів, діаграми послідовності, діаграма компонентів, був описаний функціонал як фронт-енд так і бек-енд частини системи. Для більш справедливої рейтингової системи користувачів системи було розроблено математичне забезпечення яке оцінює виконані завдання згідно досвіду того хто виконав це завдання, та штрафувало за відмову теж згідно вже накопичених очок рейтингу. Таким чином менш досвідченим помилки пробачаються, а очок рейтингу дається більше, однак і завдання вони беруть менш значимі.

Після проектування етап розробки програмного забезпечення був значно легший, адже всі діаграми вже показували функціонал обох частин системи. В ході розробки був створений програмний продукт, що відповідає всім умовам і вирішує усі поставлені задачі в етапі проектування.

Програмний застосунок було протестовано, в ході тестів якого було вирішено, що застосунок працює правильно, як і очікувалось на етапі проектування та розробки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Фомін І.Д. “РОЗРОБКА ІНФОРМАЦІЙНОЙ СИСТЕМИ ДЛЯ КОНТРОЮ СТВОРЕННЯ ТА УПРАВЛІННЯ ПРОЕКТАМИ З ВИКОРИСТАННЯМ RESTAPI” // Матеріали IV всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2020) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 24,30 квітня 2020 р. – С.40-43
2. Документація Microsoft «ASP.NET Web API»[Електронний ресурс] // Режим доступу : https://docs.microsoft.com/en-us/aspnet/web-api/
3. Документація Angular «Angular 9»[Електронний ресурс] // Режим доступу : https://angular.io/docs
4. Тепляков С.В. : Паттерны проектирования на платформе .NET. / Питер, 2015. – 320с.
5. Алекс Берсон : Client/server architecture / McGraw-Hill, Березень 29, 1996. – 569с.
6. Making HTTP Requests[Електронний ресурс] // Режим доступу https://launchschool.com/books/http/read/making_requests
7. Джон Дакетт: HTML & CSS: Design and Build Web Sites / Эксмо, 2011. – 480с.

Додаток А

Тексти програмного коду

Інформаційна система менеджменту проектів з використанням RestAPI працездатності банкоматної мережі

(Найменування програми (документа))

DVD-R

(Вид носія даних)

79 арк, 20044 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Бек-енд

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;

namespace PrjectManagmentBackEnd
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

```
using BAL.Interfaces;
using BAL.Services;
using BLL.Interfaces;
using BLL.Services;
using DAL.EF;
using DAL.Interfaces;
using DAL.UnitOfWork;
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.IdentityModel.Tokens;
using System.Text;

namespace PrjectManagmentBackEnd
{
    public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
                    .AddJwtBearer(options =>
                    {
                        options.TokenValidationParameters = new TokenValidationParameters
                        {
                            ValidateIssuer = true,
                            ValidateAudience = true,
                            ValidateLifetime = true,
                            ValidateIssuerSigningKey = true,
                            ValidIssuer = Configuration["Jwt:Issuer"],
                            ValidAudience = Configuration["Jwt:Audience"],
                            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Jwt:Key"]))
                        };
                    });


            services.AddCors(options =>
            {
                options.AddPolicy(
                  "CorsPolicy",
                  builder => builder.WithOrigins("http://localhost:4200")
                  .AllowAnyMethod()
                  .AllowAnyHeader()
                  .AllowCredentials());
            });

            services.AddDbContext<CommContext>(options => options.UseSqlServer(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\kpi\Diplom\PrjectManagmentBackEnd\Prjec
tManagmentBackEnd\AppData\ProjectManagmentDB.mdf;Integrated Security=True"));
            services.AddScoped<IUnitOfWork, UnitOfWork>();
            services.AddScoped<IProjectService, ProjectService>();
```

```
            services.AddScoped<IUserService, UserService>();
            services.AddScoped<ITaskService, TaskService>();
            services.AddScoped<IRoleService, RoleService>();
            services.AddScoped<ILinkService, LinkService>();
            services.AddControllers();
        }

        // This method gets called by the runtime. Use this method to configure the HTTP
    request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {

            app.UseAuthentication();
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

            app.UseRouting();
            app.UseCors("CorsPolicy");
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllers();
            });
        }
    }
                                            }
```

```
using System;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using BAL.DTO;
using BAL.Interfaces;
using BLL.DTO;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Configuration;
using Microsoft.IdentityModel.Tokens;

namespace PrjectManagmentBackEnd.Controllers
{
    [Route("api/[controller]")]
    [EnableCors("CorsPolicy")]
    public class LoginController : Controller
    {
        private IConfiguration _config;
        private IUserService _uservice;

        public LoginController(IUserService service, IConfiguration conf)
        {
            this._uservice = service;
            this._config = conf;
        }

        private UserDTO AuthenticateUser(UserLoginModel loginModel)
        {
            UserDTO user = _uservice.GetUserByLogin(loginModel.Login);

            if (user != null && loginModel.Password == user.Password)
            {
                return user;
            }
            return null;
        }

        private string GenerateJSONWebToken(UserDTO userInfo)
        {
            var securityKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["Jwt:Key"]));
            var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

            var claims = new[]
            {
                new Claim("ID", userInfo.UserId.ToString())
            };

            var token = new JwtSecurityToken(
                _config["Jwt:Issuer"],
                _config["Jwt:Audience"],
                claims,
                expires: DateTime.Now.AddMinutes(120),
                signingCredentials: credentials);

            return new JwtSecurityTokenHandler().WriteToken(token);
        }

        // GET: api/Login
        [AllowAnonymous]
        [HttpGet]
```

```
        public ActionResult Get()
        {
            return Ok(_uservice.GetUsers());
        }

        // POST: api/Login
        [AllowAnonymous]
        [HttpPost]
        public ActionResult Post([FromBody]UserLoginModel login)
        {
            ActionResult response = Unauthorized("Not Found");
            var user = AuthenticateUser(login);

            if (user != null)
            {
                var tokenString = GenerateJSONWebToken(user);
                response = Ok(new { token = tokenString, user = user });
            }

            return response;
        }
    }
}
```

```
using System.Collections.Generic;
using BAL.DTO;
using BAL.Interfaces;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;

namespace PrjectManagmentBackEnd.Controllers
{
    [Route("api/[controller]")]
    [EnableCors("CorsPolicy")]
    public class RoleController : Controller
    {
        protected IRoleService _service;

        public RoleController(IRoleService service)
        {
            this._service = service;
        }

        // GET: api/Role
        [HttpGet]
        public ActionResult Get()
        {
            return Ok(_service.GetRoles());
        }

        // GET: api/Role/5
        [HttpGet("{id}")]
        public ActionResult Get(int id)
        {
            return Ok(_service.GetRole(id));
        }

        // POST: api/Role
        [HttpPost]
        public ActionResult Post([FromBody] RoleDTO role)
        {
            _service.AddRole(role);
            return Ok();
        }

        // PUT: api/Role/5
        [HttpPut("{id}")]
        public ActionResult Put(int id, [FromBody] RoleDTO role)
        {
            _service.UpdateRole(id, role);
            return Ok();
        }

        // DELETE: api/ApiWithActions/5
        [HttpDelete("{id}")]
        public ActionResult Delete(int id)
        {
            _service.Delete(id);
            return Ok(true);
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using BAL.DTO;
using BAL.Interfaces;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;

namespace PrjectManagmentBackEnd.Controllers
{
    [Route("api/[controller]")]
    [EnableCors("CorsPolicy")]
    public class UserController : Controller
    {
        protected IUserService _service;

        public UserController(IUserService service)
        {
            this._service = service;
        }

        // GET: api/User/5
        [Authorize]
        [HttpGet]
        public ActionResult Get()
        {
            var id = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());
            return Ok(_service.GetUser(id));
        }

        // PUT: api/User/5
        [Authorize]
        [HttpPut]
        public ActionResult Put( [FromBody] UserDTO user)
        {
            int id = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());
            _service.UpdateUser(id, user);
            return Ok(user);
        }

        // POST: api/User/5
        [AllowAnonymous]
        [HttpPost]
        public ActionResult Post([FromBody] UserDTO user)
        {
            _service.AddUser(user);
            return Ok();
        }

        // DELETE: api/ApiWithActions/5
        [Authorize]
        [HttpDelete("{id}")]
        public ActionResult Delete(int id)
        {
            var userid = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());
            if (userid == id) {
                _service.Delete(id);
                return Ok(true);
            }
            return BadRequest(false);
```

```
}}}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using BAL.DTO;
using BAL.Interfaces;
using BLL.Interfaces;
using DAL.Entites;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.DependencyInjection;

namespace PrjectManagmentBackEnd.Controllers
{
    [Route("api/[controller]")]
    [EnableCors("CorsPolicy")]
    public class UsersProjectController : Controller
    {
        protected IProjectService _service;
        protected ILinkService _lservice;
        protected IRoleService _rservise;
        protected IUserService _uservice;
        protected ITaskService _tservice;

        public UsersProjectController(ITaskService tservice, IRoleService rservice,
IProjectService service, ILinkService lservice, IUserService uservice)
        {
            this._service = service;
            this._lservice = lservice;
            this._rservise = rservice;
            this._uservice = uservice;
            this._tservice = tservice;
        }

        // GET: api/<controller>
        [Authorize]
        [HttpGet]
        public ActionResult Get()
        {
            var id = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());

            var res = from proj in _service.GetProjects()
                      join link in _lservice.GetLinks() on proj.ProjectId equals
link.ProjectId
                      where link.UserId == id
                      join roles in _rservise.GetRoles() on link.RoleId equals
roles.RoleId
                      select new
                      {
                          ProjectId = proj.ProjectId,
                          Title = proj.title,
                          Status = proj.status,
                          Image = proj.image,
                          Discription = proj.discription,
                          Role = roles.RoleName
                      };

            return Ok(res);
        }

        [Authorize]
        [HttpGet("{Id}")]
        public ActionResult Get(int id)
```

```
        {
            string choise = HttpContext.Request.Headers
                                                    .Where(x => x.Key == "choise")
                                                    .Select(x =>
x.Value).FirstOrDefault();


            if (choise == "users") {
                var links = _lservice.GetLinks().Where(x => x.ProjectId == id).Select(x
=> x.UserId).ToList();

                var res = _uservice.GetUsers()
                    .Where(x => links.Contains(x.UserId))
                    .ToList();

                return Ok(res);
            }

            if (choise == "tasks") {
                var tasks = _tservice.GetTasks().Where(x => x.ProjectId == id).ToList();

                return Ok(tasks);
            }

            return Ok();
        }
        // POST api/<controller>
        [Authorize]
        [HttpPost]
        public ActionResult Post([FromBody]ProjectDTO project)
        {
            _service.AddProject(project);
            return Ok(project);
        }

        // PUT api/<controller>/5
        [Authorize]
        [HttpPut("{id}")]
        public ActionResult Put(int id,[FromBody]ProjectDTO project)
        {
            string choise = HttpContext.Request.Headers
                                                    .Where(x => x.Key == "choise")
                                                    .Select(x =>
x.Value).FirstOrDefault();

            if (choise == "FireUser")
            {
                int userId = Convert.ToInt32(HttpContext.Request.Headers
                                                    .Where(x => x.Key == "userid")
                                                    .Select(x =>
x.Value).FirstOrDefault());

                var linkid = _lservice.GetLinks().Where(x => x.ProjectId == id &&
x.UserId == userId).Select(x => x.Id).FirstOrDefault();
                _lservice.Delete(linkid);

                var tasks = _tservice.GetTasks().Where(x => x.ProjectId == id & x.UserId
== userId).ToList();

                foreach (var task in tasks)
                {
                    task.UserId = 0;
                    task.Status = "close";
```

```
                    _tservice.UpdateTask(task.TaskId, task);
                }
            }

            if (choise == "SetUser")
            {
                int userId = Convert.ToInt32(HttpContext.Request.Headers
                                                        .Where(x => x.Key == "userid")
                                                        .Select(x =>
x.Value).FirstOrDefault());

                var task = _tservice.GetTasks().Where(x => x.ProjectId == id & x.UserId
== userId).Select(x=> x).FirstOrDefault();

                task.UserId = userId;
                _tservice.UpdateTask(task.TaskId, task);

            }

            if (choise == "not")
            {
                _service.UpdateProject(id, project);
                return Ok(project);
            }
            return Ok();
        }

        // DELETE api/<controller>/5
        [Authorize]
        [HttpDelete("{id}")]
        public ActionResult Delete(int id)
        {
            string choise = HttpContext.Request.Headers
                                                        .Where(x => x.Key == "choise")
                                                        .Select(x =>
x.Value).FirstOrDefault();

            if (choise == "link")
            {
                var userid = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());

                var link = _lservice.GetLinks().Where(x => x.UserId == userid &&
x.ProjectId == id).FirstOrDefault();
                _lservice.Delete(link.Id);
            }
            if (choise == "project")
            {
                var links = _lservice.GetLinks().Where(x => x.ProjectId == id).ToList();

                foreach (var l in links)
                {
                    _lservice.Delete(l.Id);
                }

                _service.Delete(id);
            }

            return Ok(true);
        }
    }
}
```

```
using System.Collections.Generic;
using BAL.DTO;
using BAL.Interfaces;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using System.Linq;
using Microsoft.AspNetCore.Authorization;
using System;

namespace PrjectManagmentBackEnd.Controllers
{
    [Route("api/[controller]")]
    [Authorize]
    [EnableCors("CorsPolicy")]
    public class UserTaskController : Controller
    {
        protected ITaskService _tservice;
        protected IUserService _uservice;
        protected IProjectService _pservice;

        public UserTaskController(ITaskService tservice, IUserService uservise,
IProjectService pservice)
        {
            this._tservice = tservice;
            this._uservice = uservise;
            this._pservice = pservice;
        }
        // GET: api/Task/Id
        [Authorize]
        [HttpGet("{Id}")]
        public ActionResult Get(int Id)
        {
            var userid = Convert.ToInt32(HttpContext.User.Claims.Where(c => c.Type ==
"ID").Select(c => c.Value).FirstOrDefault());

            var result = from task in _tservice.GetTasks()
                         join project in _pservice.GetProjects() on task.ProjectId equals
project.ProjectId
                         where project.ProjectId == Id
                         join user in _uservice.GetUsers() on task.UserId equals
user.UserId
                         where user.UserId == userid
                         select new
                         {
                             TaskId = task.TaskId,
                             UserId = task.UserId,
                             UserName = user.UserName,
                             ProjectId = task.ProjectId,
                             ProjectTitle = project.title,
                             TaskParentID = task.ParentTaskId,
                             TaskName = task.TaskName,
                             TaskDiscritpion = task.Disription,
                             TaskStatus = task.Status,
                             TaskCreateDate = task.CreateDate,
                             RatingPoints = task.RatingPoints,
                             TaskEstimeteDate = task.EstimateDate
                         };
            return Ok(result);
        }

        // POST: api/Task
        [HttpPost]
        public ActionResult Post([FromBody] TaskDTO task)
        {
```

```
            _tservice.AddTask(task);
            return Ok();
        }

        // PUT: api/Task/5
        [HttpPut("{id}")]
        public ActionResult Put(int id, [FromBody] TaskDTO task)
        {
            _tservice.UpdateTask(id, task);
            return Ok();
        }

        // DELETE: api/ApiWithActions/5
        [HttpDelete("{id}")]
        public ActionResult Delete(int id)
        {
            _tservice.Delete(id);
            return Ok(true);
        }
    }
}
```

```
using BAL.DTO;
using System;
using System.Collections.Generic;
using System.Text;

namespace BLL.Interfaces
{
    public interface ILinkService
    {
        void AddLink(ProjectUserRoleDTO link);
        ProjectUserRoleDTO GetLink(int? id);
        List<ProjectUserRoleDTO> GetLinks();
        void UpdateLink(int? id, ProjectUserRoleDTO link);
        bool Delete(int? id);
        void Dispose();
    }
}
```

```
using BAL.DTO;
using System.Collections.Generic;

namespace BAL.Interfaces
{
    public interface IProjectService
    {
        void AddProject(ProjectDTO project);
        ProjectDTO GetProject(int? id);
        List<ProjectDTO> GetProjects();
        void UpdateProject(int? id, ProjectDTO project);
        bool Delete(int? id);
        void Dispose();
    }
}
```

```
using BAL.DTO;
using System.Collections.Generic;

namespace BAL.Interfaces
{
    public interface IRoleService
    {
        void AddRole(RoleDTO role);
        RoleDTO GetRole(int? id);
        List<RoleDTO> GetRoles();
        void UpdateRole(int? id, RoleDTO role);
        bool Delete(int? id);
        void Dispose();
    }
}
```

```
using BAL.DTO;
using System.Collections.Generic;

namespace BAL.Interfaces
{
    public interface ITaskService
    {
        void AddTask(TaskDTO task);
        TaskDTO GetTask(int? id);
        List<TaskDTO> GetTasks();
        void UpdateTask(int? id, TaskDTO task);
        bool Delete(int? id);
        void Dispose();
    }
}
```

```
using BAL.DTO;
using System.Collections.Generic;

namespace BAL.Interfaces
{
    public interface IUserService
    {
        void AddUser(UserDTO user);
        UserDTO GetUser(int? id);
        UserDTO GetUserByLogin(string login);
        List<UserDTO> GetUsers();
        void UpdateUser(int? id, UserDTO user);
        bool Delete(int? id);
        void Dispose();
    }
}
```

```
using AutoMapper;
using BAL.DTO;
using BLL.Interfaces;
using DAL.Entites;
using DAL.Interfaces;
using System;
using System.Collections.Generic;
using System.Text;

namespace BLL.Services
{
    public class LinkService : ILinkService
    {
        IUnitOfWork Database { get; set; }

        public LinkService(IUnitOfWork db)
        {
            this.Database = db;
        }

        public void AddLink(ProjectUserRoleDTO link)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<ProjectUserRoleDTO,
ProjectUserRole>()).CreateMapper();

            Database.Links.Create(mapper.Map<ProjectUserRoleDTO, ProjectUserRole>(link));
            Database.Save();
        }

        public bool Delete(int? id)
        {
            ProjectUserRole link = Database.Links.Get(id.Value);
            if (link != null)
            {
                Database.Links.Delete(id.Value);
                Database.Save();
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public ProjectUserRoleDTO GetLink(int? id)
        {
            ProjectUserRole link = Database.Links.Get(id.Value);
            if (link != null)
            {
                var mapper = new MapperConfiguration(cfg =>
cfg.CreateMap<ProjectUserRole, ProjectUserRoleDTO>()).CreateMapper();
                return mapper.Map<ProjectUserRole, ProjectUserRoleDTO>(link);
            }
            else
            {
                throw new Exception("Not Found!");
            }
        }
```

```
        public List<ProjectUserRoleDTO> GetLinks()
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<ProjectUserRole,
ProjectUserRoleDTO>()).CreateMapper();
            return mapper.Map<IEnumerable<ProjectUserRole>,
List<ProjectUserRoleDTO>>(Database.Links.GetAll());
        }

        public void UpdateLink(int? id, ProjectUserRoleDTO link)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<ProjectUserRoleDTO,
ProjectUserRole>()).CreateMapper();
            this.Database.Links.Update(mapper.Map<ProjectUserRoleDTO,
ProjectUserRole>(link));
            this.Database.Save();
        }
    }
}
```

```
using BAL.DTO;
using BAL.Interfaces;
using System;
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using AutoMapper;

namespace BAL.Services
{
    public class ProjectService : IProjectService
    {
        IUnitOfWork Database { get; set; }

        public ProjectService(IUnitOfWork db)
        {
            this.Database = db;
        }

        public void AddProject(ProjectDTO projectdto)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<ProjectDTO,
Project>()).CreateMapper();

            Database.Projects.Create(mapper.Map<ProjectDTO, Project>(projectdto));
            Database.Save();
        }

        public bool Delete(int? id)
        {
            Project project = Database.Projects.Get(id.Value);
            if (project != null)
            {
                Database.Projects.Delete(id.Value);
                Database.Save();
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public ProjectDTO GetProject(int? id)
        {
            Project project = Database.Projects.Get(id.Value);
            if (project != null)
            {
                var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Project,
ProjectDTO>()).CreateMapper();
                return mapper.Map<Project, ProjectDTO>(project);
            }
            else
            {
                throw new Exception("Not Found!");
            }
        }

        public List<ProjectDTO> GetProjects()
```

```
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Project,
ProjectDTO>()).CreateMapper();
            return mapper.Map<IEnumerable<Project>,
List<ProjectDTO>>(Database.Projects.GetAll());
        }

        public void UpdateProject(int? id, ProjectDTO project)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<ProjectDTO,
Project>()).CreateMapper();
            this.Database.Projects.Update(mapper.Map<ProjectDTO, Project>(project));
            this.Database.Save();
        }
    }
}
```

```
using AutoMapper;
using BAL.DTO;
using BAL.Interfaces;
using DAL.Entites;
using DAL.Interfaces;
using System;
using System.Collections.Generic;

namespace BAL.Services
{
    public class RoleService : IRoleService
    {
        IUnitOfWork Database { get; set; }

        public RoleService(IUnitOfWork db)
        {
            this.Database = db;
        }

        public void AddRole(RoleDTO roledto)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<RoleDTO,
Role>()).CreateMapper();

            Database.Roles.Create(mapper.Map<RoleDTO, Role>(roledto));
            Database.Save();
        }

        public bool Delete(int? id)
        {
            Role role = Database.Roles.Get(id.Value);
            if (role != null)
            {
                Database.Roles.Delete(id.Value);
                Database.Save();
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public RoleDTO GetRole(int? id)
        {
            Role role = Database.Roles.Get(id.Value);
            if (role != null)
            {
                var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Role,
RoleDTO>()).CreateMapper();
                return mapper.Map<Role, RoleDTO>(role);
            }
            else
            {
                throw new Exception("Not Found!");
            }
        }

        public List<RoleDTO> GetRoles()
```

```
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Role,
RoleDTO>()).CreateMapper();
            return mapper.Map<IEnumerable<Role>, List<RoleDTO>>(Database.Roles.GetAll());
        }

        public void UpdateRole(int? id, RoleDTO role)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<RoleDTO,
Role>()).CreateMapper();
            this.Database.Roles.Update(mapper.Map<RoleDTO, Role>(role));
            this.Database.Save();
        }
    }
}
```

```
using AutoMapper;
using BAL.DTO;
using BAL.Interfaces;
using DAL.Entites;
using DAL.Interfaces;
using System;
using System.Collections.Generic;

namespace BAL.Services
{
    public class RoleService : IRoleService
    {
        IUnitOfWork Database { get; set; }

        public RoleService(IUnitOfWork db)
        {
            this.Database = db;
        }

        public void AddRole(RoleDTO roledto)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<RoleDTO,
Role>()).CreateMapper();

            Database.Roles.Create(mapper.Map<RoleDTO, Role>(roledto));
            Database.Save();
        }

        public bool Delete(int? id)
        {
            Role role = Database.Roles.Get(id.Value);
            if (role != null)
            {
                Database.Roles.Delete(id.Value);
                Database.Save();
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public RoleDTO GetRole(int? id)
        {
            Role role = Database.Roles.Get(id.Value);
            if (role != null)
            {
                var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Role,
RoleDTO>()).CreateMapper();
                return mapper.Map<Role, RoleDTO>(role);
            }
            else
            {
                throw new Exception("Not Found!");
            }
        }

        public List<RoleDTO> GetRoles()
```

```
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<Role,
RoleDTO>()).CreateMapper();
            return mapper.Map<IEnumerable<Role>, List<RoleDTO>>(Database.Roles.GetAll());
        }

        public void UpdateRole(int? id, RoleDTO role)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<RoleDTO,
Role>()).CreateMapper();
            this.Database.Roles.Update(mapper.Map<RoleDTO, Role>(role));
            this.Database.Save();
        }
    }
}
```

```
using AutoMapper;
using BAL.DTO;
using BAL.Interfaces;
using DAL.Entites;
using DAL.Interfaces;
using System;
using System.Collections.Generic;

namespace BAL.Services
{
    public class UserService : IUserService
    {
        IUnitOfWork Database { get; set; }

        public UserService(IUnitOfWork db)
        {
            this.Database = db;
        }

        public void AddUser(UserDTO userdto)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<UserDTO,
User>()).CreateMapper();

            Database.Users.Create(mapper.Map<UserDTO, User>(userdto));
            Database.Save();
        }

        public bool Delete(int? id)
        {
            User user = Database.Users.Get(id.Value);
            if (user != null)
            {
                Database.Users.Delete(id.Value);
                Database.Save();
                return true;
            }
            else
            {
                return false;
            }
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public UserDTO GetUser(int? id)
        {
            User user = Database.Users.Get(id.Value);
            if (user != null)
            {
                var mapper = new MapperConfiguration(cfg => cfg.CreateMap<User,
UserDTO>()).CreateMapper();
                return mapper.Map<User, UserDTO>(user);
            }
            else
            {
                throw new Exception("Not Found!");
            }
        }

        public List<UserDTO> GetUsers()
```

```
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<User,
UserDTO>()).CreateMapper();
            return mapper.Map<IEnumerable<User>, List<UserDTO>>(Database.Users.GetAll());
        }

        public void UpdateUser(int? id, UserDTO user)
        {
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<UserDTO,
User>()).CreateMapper();
            this.Database. Users.Update(mapper.Map<UserDTO, User>(user));
            this.Database.Save();
        }

        public UserDTO GetUserByLogin(string login)
        {
            User user = Database.Users.GetByLogin(login);
            var mapper = new MapperConfiguration(cfg => cfg.CreateMap<User,
UserDTO>()).CreateMapper();
            return mapper.Map<User, UserDTO>(user);
        }
    }
                                                }
```

```
namespace BAL.DTO
{
    public class ProjectDTO
    {
        public int ProjectId { get; set; }
        public string title { get; set; }

        public string status { get; set; }

        public string image { get; set; }
        public string discription { get; set; }
    }
}
```

```
namespace BAL.DTO
{
    public class ProjectUserRoleDTO
    {
        public int Id { get; set; }
        public int RoleId { get; set; }
        public int ProjectId { get; set; }
        public int UserId { get; set; }

    }
}
```

```
namespace BAL.DTO
{
    public class RoleDTO
    {
        public int RoleId { get; set; }
        public string RoleName { get; set; }
        public int Priority { get; set; }
    }
}
```

```
using System;

namespace BAL.DTO
{
    public class TaskDTO
    {
        public int TaskId { get; set; }
        public int UserId { get; set; }

        public int ProjectId { get; set; }

        public int ParentTaskId { get; set; }

        public string TaskName { get; set; }

        public string Disription { get; set; }
        public string Status { get; set; }
        public int RatingPoints { get; set; }
        public DateTime CreateDate { get; set; }
        public DateTime EstimateDate { get; set; }
    }
}
```

```
using Microsoft.AspNetCore.Identity;
using System;

namespace BAL.DTO
{
    public class UserDTO
    {
        public int UserId { get; set; }
        public string Login { get; set; }

        public string Password { get; set; }

        public string UserName { get; set; }
        public int Finished { get; set; }
        public int Failed { get; set; }
        public int Expired { get; set; }
        public double Rating { get; set; }
    }
}
```

```
using Microsoft.AspNetCore.Identity;
using System;
using System.Collections.Generic;
using System.Text;

namespace BLL.DTO
{
    public class UserLoginModel : IdentityUser
    {
        public string Login { get; set; }

        public string Password { get; set; }
    }
}
```

```
using DAL.Entites;
using Microsoft.EntityFrameworkCore;

namespace DAL.EF
{
    public class CommContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<Project> Projects { get; set; }
        public DbSet<ProjectUserRole> ProjectUserRole { get; set; }
        public DbSet<Task> Taskss { get; set; }
        public DbSet<Role> Roles { get; set; }


        public CommContext(DbContextOptions<CommContext> options) : base(options)
        {
            Database.EnsureCreated();
        }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<Task>()
                .HasMany(oj => oj.ChildTasks)
                .WithOne(j => j.ParentTask)
                .HasForeignKey(j => j.TaskId);
        }

    }
}
```

```
using System.Collections.Generic;

namespace DAL.Entites
{
    public class Project
    {
        public int ProjectId {get;set;}
        public string title { get; set; }

        public string status { get; set; }

        public string image { get; set; }
        public string discription { get; set; }

        public ICollection<Task> Tasks { get; set; }

        public ICollection<ProjectUserRole> projectUserRoles { get; set; }

        public Project()
        {
            this.projectUserRoles = new List<ProjectUserRole>();
            this.Tasks = new List<Task>();
        }
    }
}
```

```
using System.ComponentModel.DataAnnotations;

namespace DAL.Entites
{
    public class ProjectUserRole
    {
        [Key]
        public int Id { get; set; }
        public int RoleId { get; set; }
        public int ProjectId { get; set; }
        public int UserId { get; set; }
    }
}
```

```
using System.Collections.Generic;

namespace DAL.Entites
{
    public class Role
    {
        public int RoleId { get; set; }

        public string RoleName { get; set; }
        public int Priority { get; set; }

        public ICollection<ProjectUserRole> projectUserRoles { get; set; }

        public Role()
        {
            this.projectUserRoles = new List<ProjectUserRole>();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DAL.Entites
{
    public class Task
    {
        [Key]
        public int TaskId { get; set; }

        [ForeignKey("FK_Taskss_ToUser")]
        public int UserId { get; set; }
        [ForeignKey("FK_Taskss_ToProject")]
        public int ProjectId { get; set; }
        [ForeignKey("FK_Taskss_ToTaskss")]
        public int? ParentTaskId { get; set; }

        public string TaskName { get; set; }

        public string Disription { get; set; }
        public string Status { get; set; }
        public DateTime CreateDate { get; set; }
        public DateTime EstimateDate { get; set; }

        public int RatingPoints { get; set; }

        public Task ParentTask { get; set; }
        public Project Project { get; set; }

        public User User { get; set; }
        public ICollection<Task> ChildTasks { get; set; }

        public Task()
        {
            this.ChildTasks = new List<Task>();
        }
    }
}
```

```
using System;
using System.Collections.Generic;

namespace DAL.Entites
{
    public class User
    {
        public int UserID { get; set; }

        public string Login { get; set; }

        public string Password { get; set; }

        public string UserName { get; set; }
        public int Finished { get; set; }
        public int Failed { get; set; }
        public int Expired { get; set; }
        public double Rating { get; set; }

        public ICollection<Task> Tasks { get; set; }

        public ICollection<ProjectUserRole> projectUserRoles { get; set; }

        public User()
        {
            this.projectUserRoles = new List<ProjectUserRole>();
            this.Tasks = new List<Task>();
        }
    }
}
```

```
using System.Collections.Generic;

namespace DAL.Interfaces
{
    public interface IRepository<T> where T : class
    {
        IEnumerable<T> GetAll();
        T Get(int id);
        void Create(T item);
        void Update(T item);
        void Delete(int id);
    }
}
```

```
using DAL.Repositories;
using System;
using System.Collections.Generic;
using System.Text;

namespace DAL.Interfaces
{
    public interface IUnitOfWork
    {
        public LinkRepository Links { get; }
        public ProjectRepository Projects { get; }
        public RoleRepository Roles { get; }
        public TaskRepository Tasks { get; }
        public UserRepository Users { get; }

        public void Save();
        public void Dispose();
    }
}
```

```
using System;
using DAL.EF;
using DAL.Interfaces;
using DAL.Repositories;

namespace DAL.UnitOfWork
{
    public class UnitOfWork : IDisposable, IUnitOfWork
    {
        private CommContext db;
        private LinkRepository linkRepository;
        private ProjectRepository projectRepository;
        private RoleRepository roleRepository;
        private TaskRepository taskRepository;
        private UserRepository userRepository;

        public UnitOfWork(CommContext dbc)
        {
            this.db = dbc;
        }

        public LinkRepository Links
        {
            get
            {
                if (linkRepository == null)
                    linkRepository = new LinkRepository(db);
                return linkRepository;
            }
        }

        public ProjectRepository Projects
        {
            get
            {
                if (projectRepository == null)
                    projectRepository = new ProjectRepository(db);
                return projectRepository;
            }
        }

        public RoleRepository Roles
        {
            get
            {
                if (roleRepository == null)
                    roleRepository = new RoleRepository(db);
                return roleRepository;
            }
        }

        public TaskRepository Tasks
        {
            get
            {
                if (taskRepository == null)
                    taskRepository = new TaskRepository(db);
                return taskRepository;
            }
        }

        public UserRepository Users
        {
            get
```

```
            {
                if (userRepository == null)
                    userRepository = new UserRepository(db);
                return userRepository;
            }
        }

        public void Save()
        {
            db.SaveChanges();
        }

        private bool disposed = false;
        public virtual void Dispose(bool disposing)
        {
            if (!this.disposed)
            {
                if (disposing)
                {
                    db.Dispose();
                }
                this.disposed = true;
            }
        }
        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }
    }
}
```

```
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using DAL.EF;
using Microsoft.EntityFrameworkCore;

namespace DAL.Repositories
{
    public class LinkRepository : IRepository<ProjectUserRole>
    {
        private CommContext db;

        public LinkRepository(CommContext ctx)
        {
            this.db = ctx;
        }

        public void Create(ProjectUserRole link)
        {
            db.ProjectUserRole.Add(link);
        }

        public void Delete(int id)
        {
            ProjectUserRole link = db.ProjectUserRole.Find(id);
            if (link != null)
                db.ProjectUserRole.Remove(link);
        }

        public ProjectUserRole Get(int id)
        {
            return db.ProjectUserRole.Find(id);
        }

        public IEnumerable<ProjectUserRole> GetAll()
        {
            return db.ProjectUserRole;
        }

        public void Update(ProjectUserRole link)
        {
            db.Entry(link).State = EntityState.Modified;
        }
    }
}
```

```
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using DAL.EF;
using Microsoft.EntityFrameworkCore;

namespace DAL.Repositories
{
    public class ProjectRepository : IRepository<Project>
    {
        private CommContext db;

        public ProjectRepository(CommContext ctx)
        {
            this.db = ctx;
        }

        public void Create(Project project)
        {
            db.Projects.Add(project);
        }

        public void Delete(int id)
        {
            Project project = db.Projects.Find(id);
            if (project != null)
                db.Projects.Remove(project);
        }

        public Project Get(int id)
        {
            return db.Projects.Find(id);
        }

        public IEnumerable<Project> GetAll()
        {
            return db.Projects;
        }

        public void Update(Project project)
        {
            db.Entry(project).State = EntityState.Modified;
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using DAL.EF;
using Microsoft.EntityFrameworkCore;

namespace DAL.Repositories
{
    public class RoleRepository : IRepository<Role>
    {
        private CommContext db;

        public RoleRepository(CommContext ctx)
        {
            this.db = ctx;
        }

        public void Create(Role role)
        {
            db.Roles.Add(role);
        }

        public void Delete(int id)
        {
            Role role = db.Roles.Find(id);
            if (role != null)
                db.Roles.Remove(role);
        }

        public Role Get(int id)
        {
            return db.Roles.Find(id);
        }

        public IEnumerable<Role> GetAll()
        {
            return db.Roles;
        }

        public void Update(Role role)
        {
            db.Entry(role).State = EntityState.Modified;
        }
    }
}
```

```
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using DAL.EF;
using Microsoft.EntityFrameworkCore;

namespace DAL.Repositories
{
    public class TaskRepository : IRepository<Task>
    {
        private CommContext db;

        public TaskRepository(CommContext ctx)
        {
            this.db = ctx;
        }

        public void Create(Task task)
        {
            db.Taskss.Add(task);
        }

        public void Delete(int id)
        {
            Task task = db.Taskss.Find(id);
            if (task != null)
                db.Taskss.Remove(task);
        }

        public Task Get(int id)
        {
            return db.Taskss.Find(id);
        }

        public IEnumerable<Task> GetAll()
        {
            return db.Taskss;
        }

        public void Update(Task task)
        {
            db.Entry(task).State = EntityState.Modified;
        }
    }
}
```

```
using System.Collections.Generic;
using DAL.Interfaces;
using DAL.Entites;
using DAL.EF;
using Microsoft.EntityFrameworkCore;
using System.Linq;

namespace DAL.Repositories
{
    public class UserRepository : IRepository<User>
    {
        private CommContext db;

        public UserRepository(CommContext ctx)
        {
            this.db = ctx;
        }

        public void Create(User user)
        {
            db.Users.Add(user);
        }

        public void Delete(int id)
        {
            User user = db.Users.Find(id);
            if (user != null)
                db.Users.Remove(user);
        }

        public User Get(int id)
        {
            return db.Users.Find(id);
        }

        public User GetByLogin(string login)
        {
            return db.Users.Where(x => x.Login == login).FirstOrDefault();
        }

        public IEnumerable<User> GetAll()
        {
            return db.Users;
        }

        public void Update(User user)
        {
            db.Entry(user).State = EntityState.Modified;
        }
    }
}
```

Фронт-енд

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http'
import { FormsModule } from '@angular/forms'

import { AppComponent } from './app.component';
import { ProjectComponent } from './UsersProject/usersproject.component';
import { UserTaskComponent } from './UserTask/usertask.component';
import { UserComponent } from './User/user.component';
import { RoleComponent } from './Role/role.component';
import { LoginComponent } from './Login/LoginComponent';
import { RegisterComponent} from './Register/RegisterComponent';
import { ProfileComponent } from './Profile/profile.component';
import {AllProjectsComponent} from './AllProjects/allpojects.component'

@NgModule({
  declarations: [
    AppComponent,
    ProjectComponent,
    UserTaskComponent,
    UserComponent,
    RoleComponent,
    LoginComponent,
    RegisterComponent,
    ProfileComponent,
    AllProjectsComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
import { Component, OnInit} from '@angular/core';
import { Project } from './Models/Project';
import { Task } from './Models/Task';
import { User } from './Models/User';
import { Role } from './Models/Role';
import { Puttask } from './Models/puttask'
import { HttpClient, HttpHeaders } from '@angular/common/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {
  private url='https://localhost:44312/api/';
  title = 'ClientApp';
  user: User = new User();
  token: string;
  selectedProject: number;
  usersprojects: Project[];
  selectTask: string = "None";
  selectData: Array<string> = [ "ProjectTasks", "MyTasks"]

  profile: boolean = false;
  showProjects: boolean = false;
  logedIn: boolean = false;
  login: boolean = true;
  register: boolean = false;
  showusertask: boolean = false;

  constructor(private cilent: HttpClient) {}

  ngOnInit(): void {
    this.token = localStorage.getItem("auth_token");
    if(this.token != null && this.token != ""){
      const headrDict = {
        'Content-Type': 'application/json',
        'Accept': 'application/json',
        'Access-Control-Allow-Headers': 'Content-Type',
        'Authorization': 'Bearer '+ this.token
      }

      const requestOption = {
        headers: new HttpHeaders(headrDict)
      }

      this.cilent.get(this.url+'User', requestOption).subscribe((data: User) => t
his.user = data)
      this.LogedIn(true)
    }
```

```
  }

  SetProjectId(v: number) {
    this.selectedProject = v;
    this.showusertask = false;
  }

  ShowProjects(): void{
    this.Ofer();
    this.showProjects = true;
  }

  CloseTasks():void{
    this.showusertask = false;
  }

  ShowProfile(): void{
    this.Ofer();
    this.profile = true;
  }

  ShowTasks(v: boolean):void {
    this.showusertask = true;
  }

  Ofer(){
    this.profile = false
    this.showProjects = false
    this.login = false
    this.register = false
    this.showusertask = false;
  }

  LogedIn(vr: boolean): void{
    const headrDict = {
      'Content-Type': 'application/json',
      'Accept': 'application/json',
      'Access-Control-Allow-Headers': 'Content-Type',
      'Authorization': 'Bearer '+ this.token
    }

    const requestOption = {
      headers: new HttpHeaders(headrDict)
    }


    this.cilent.get(this.url+'UsersProject', requestOption).subscribe((data: Proj
ect[])=> this.usersprojects = data);
    this.Ofer();
    this.logedIn = true;
```

```
    this.ShowProfile();
  }

  Login(): void{
    this.Ofer();
    this.login = true;
  }

  SetUser(us: User) {
      this.user = us;
  }

  Register(): void {
    this.Ofer();
    this.register = true;
  }

  LogOff():void {
    this.logedIn = false;
    this.Ofer();
    this.login = true;
  }

  setToken(tkn: string): void {
    this.token = tkn;
  }
}
```

```
.appTable {
    width: 100%;
}

.profile {
    width: 49%;
    position: absolute;
    top: 15%;
    left: 15px;
}

.projects {
    width: 49%;
    position: absolute;
    top: 60%;
    left: 15px;
}

.tasks {
    width: 49%;
    position: absolute;
    top: 18%;
    right: 15px;
}
```

```
<html>
<body>
    <header class = "header1">
      <nav>
        <ul>
          <li class="li1"> Project Management </li>
        </ul>
      </nav>
      <input *ngIf="!logedIn" type="button" value="Login" class="navBut"  style="
margin-left: 50%;"  (click) = "Login()">
      <input *ngIf="!logedIn" type="button" value="Register" class="navBut" (clic
k) = "Register()">
      <input *ngIf="logedIn" type="button" value="Projects" class="navBut" style=
"margin-left: 50%;" (click) = "ShowProjects()">
      <input *ngIf="logedIn" type="button" value="Profile" class="navBut"   (clic
k) = "ShowProfile()">
    </header>
    <login *ngIf="login" (resp)="LogedIn($event)" (setToken)="setToken($event)" (
setUser)="SetUser($event)" ></login>
    <register *ngIf="register" (resp)="Login()" ></register>
    <div *ngIf="logedIn">
       <profile class="profile" [user] = "user" [token] = "token" (logoff) = "Log
Off()"> </profile>
      <h3 *ngIf="!showProjects" style="position: absolute; top: 16%; right: 15px;
"> *To check your or available tasks, please select Project and push Check Tasks
button </h3>
      <usertask  class="tasks" *ngIf="showusertask" [token] = "token" [projectId]
 = "selectedProject" (close) = "CloseTasks()"> </usertask>
      <h1 *ngIf="!showProjects" style="position: absolute; top: 50%"> Below you c
an check project you are involved in! </h1>
      <usersproject class="projects"  *ngIf="profile" (tasks)="ShowTasks($event)"
 (projid)="SetProjectId($event)" [token] ="token" [projects] = "usersprojects">
</usersproject>
      <h1 *ngIf="showProjects" style="position: absolute; top: 50%"> Below you ca
n check all projects in Project Management system! </h1>
      <allp *ngIf="showProjects"> </allp>
    </div>
</body>
</html>
```

```
import {Component, Input, OnInit, Output, EventEmitter} from '@angular/core'
import { Task } from '../Models/Task';
import { Puttask } from '../Models/puttask'
import { HttpClient, HttpHeaders } from '@angular/common/http'

@Component({
    selector: 'usertask',
    templateUrl: './usertask.component.html',
    styleUrls: ['./usertask.component.css']
})
export class UserTaskComponent implements OnInit {
    @Input() projectId: number;
    @Input() token: string;
    @Output() close = new EventEmitter();
    selectTask: string = null;
    tasks: Task[];


    userTasks: boolean = true;
    private url='https://localhost:44312/api/UserTask';

    constructor (private client: HttpClient) {}

    ngOnInit(): void {

    }

    onClose(): void {
        this.close.emit();
    }

    onChange(v: string): void {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }
        if(v == "projectTasks") {

        }
        if(v == "usertasks") {
            this.client.get(this.url+'/'+this.projectId, requestOption).subscribe
((data: Task[])=> this.tasks = data );
        }
    }
```

```
}
.back {
    background-color: rgb(54, 90, 109);
    width: 100%;
    margin-top: 50px;
    font-family: Georgia, 'Times New Roman', Times, serif;
    font-size: medium;
}

.lis {
    margin-left: 80%;
    height: 40px;
    width: 20%;
    margin-top: 50px;
    position: static;
    align-items: center;
    font-family: Georgia, 'Times New Roman', Times, serif;
    font-size: medium;
}

.but{
    height: 40px;
    width: 10%;
    background-color: rgb(98, 98, 230);
    margin-left: 90%;
}
```

```
<div>
    <select [(ngModel)] = selectTask #MySelect name="tasks" class="lis" (change)=
"onChange(MySelect.value)" required> <option [ngValue]="null" [disabled]="true" >
 Select type of tasks </option> <option value="projectTasks"> Project Tasks </opt
ion> <option value="usertasks"> User Tasks </option> </select>
    <div class="back"  *ngFor="let task of tasks">
        <p> {{task.taskName}} </p>
        <p> {{task.taskDiscritpion}} </p>
        <p> For finishing this task you will earn {{task.ratingPoints}} points <p
>
        <p> Status: {{task.taskStatus}} </p>
        <p> Createted on {{task.taskCreateDate | date:'medium' }} </p>
        <p> Will be expired on {{task.taskEstimeteDate | date:'medium'}} </p>
        <input type="button" value="Leave" class="but">
        <input type="button" value="Finish" class="but">
    </div>
    <input *ngIf="selectTask" class="but" type="button" value="Close tasks" (clic
k)="onClose()">
</div>
```

```
import {Component, Input, OnInit, Output, EventEmitter} from '@angular/core'
import { Project } from '../Models/Project';
import { HttpClient, HttpHeaders } from '@angular/common/http'
import { Task } from '../Models/Task';
import { User } from '../Models/User';

@Component({
    selector: 'usersproject',
    templateUrl: './usersproject.component.html',
    styleUrls: ['./userprojects.component.css']
})
export class ProjectComponent implements OnInit {
    @Output() projid = new EventEmitter();
    @Output() tasks = new EventEmitter();
    @Input()projects: Project[];
    @Input() token: string;
    projecttasks: Task[];
    projectusers: User[];
    selectedTask: Task = null;
    secetedUser: User = null;
    selectedUser: User = null;
    selectedTaskk: string = null;
    secetedUserr: string = null;
    selectedUserr: string = null;
    project: Project= new Project();
    deleted: boolean = false;
    manageu: boolean = false;
    managet: boolean = false;
    private url='https://localhost:44312/api/UsersProject';


    constructor (private client: HttpClient) {}

    ngOnInit(): void { }

    SelectTask(i: number) {
        this.selectedTask = this.projecttasks.find(x=> x.taskId == i);
    }

    SelectUser(i: number) {
        this.secetedUser = this.projectusers.find(x=>x.userId == i);
    }

    SelectTaskUser(i: number) {
        this.selectedUser = this.projectusers.find(x=>x.userId == i);
    }

    Delete(id: number): void {
        const headrDict = {
            'Content-Type': 'application/json',
```

```
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'project',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }

        this.client.delete(this.url+'/' + id, requestOption).subscribe( (data: bo
olean) => this.deleted = data)
    }

    Save(i:number): void {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'not',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }
        this.client.put(this.url + '/' + this.projects[i].projectId, this.project
s[i],requestOption).subscribe((data: Project[]) => this.projects)
    }

    Cancel():void {
        this.project = null;
        this.selectedTask = null;
        this.selectedTaskk = null;
        this.selectedUser = null;
        this.selectedUserr = null;
        this.secetedUser = null;
        this.secetedUserr = null;
        this.manageu = false;
        this.managet = false;
    }

    CheckTask(projid: number): void {
        this.projid.emit(projid);
        this.tasks.emit(true);
    }

    Leave(projId: number):void {
        const headrDict = {
            'Content-Type': 'application/json',
```

```
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'link',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }

        this.client.delete(this.url+'/'+projId,requestOption).subscribe((data: bo
olean)=> this.deleted = true);
    }

    ManageUsers():void {
        this.selectedTask = null;
        this.selectedTaskk = null;
        this.selectedUser = null;
        this.selectedUserr = null;
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'users',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }

        this.client.get(this.url+'/'+this.project.projectId, requestOption).subsc
ribe( (data: User[]) => this.projectusers = data )

        this.manageu = true;
        this.managet = false;
    }

    ManageTasks():void {
        this.secetedUser = null;
        this.secetedUserr = null;
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'tasks',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
```

```
            headers: new HttpHeaders(headrDict)
        }

        this.client.get(this.url+'/'+this.project.projectId, requestOption).subsc
ribe( (data: Task[]) => this.projecttasks = data )

        this.manageu = false;
        this.managet = true;
    }

    FireUser():void {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'FireUser',
            'UserId': ''+this.secetedUser.userId,
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }

        this.client.put(this.url+'/'+this.project.projectId, this.project, reques
tOption).subscribe((data: Project) => this.project = data);
    }

    ChangeUserOnTask(): void {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'choise': 'SetUser',
            'UserId': ''+this.selectedUser.userId,
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }

        this.client.put(this.url+'/'+this.project.projectId, this.project, reques
tOption).subscribe((data: Project) => this.project = data);
    }

    Update(proj: Project) {
        this.project = proj;
    }
}
```

```
<div class="back" *ngFor="let projec of projects; let i = index">
    <ng-
template [ngIf]="project?.projectId != projec.projectId" [ngIfElse]="edit">
    <img class= "mage" *ngIf="projects[i].image" [src]="projects[i].image">
    <h1> {{projects[i].title}} </h1>
    <h2> {{projects[i].discription}} </h2>
    <h2> {{projects[i].status}} </h2>
    <p style="margin-
left: 73%;"> Your role on this project : {{projects[i].role}}</p>
    <input style="margin-
left: 67%;" class="but" type="button" value="Check Tasks" (click) = "CheckTask(pr
ojec.projectId)">
    <input class="but" type="button" value="Leave Project" (click) = "Leave(proje
c.projectId)"> <br>
    <input style="margin-
left: 67%;" *ngIf="projec.role == 'Boss'" class="but" type="button" value="Delete
 Project" (click) = "Delete(projec.projectId)">
    <input *ngIf="projec.role == 'Boss'" class="but" type="button" value="Update
Project" (click) = "Update(projec)">
    </ng-template>
    <ng-template #edit>
        <input class="tex" type="text" placeholder="Title" [(ngModel)]="projects[
i].title" >
        <select class="sec" *ngIf="manageu" #MySelect1 [(ngModel)]="secetedUserr"
 (change)="SelectUser(MySelect1.value)" required>
            <option [ngValue]="null" [disabled]="true"> Select User </option>
            <option *ngFor="let user of projectusers" [value]="user.userId"> {{us
er.userName}} </option>
        </select>
        <select class="sec" *ngIf="managet" #MySelect2 [(ngModel)]="selectedTaskk
" (change)="SelectTask(MySelect2.value)" required>
            <option [ngValue]="null" [disabled]="true"> Select Task </option>
            <option *ngFor="let task of projecttasks" [value]="task.taskId"> {{ta
sk.taskName}} </option>
        </select>
        <br>
        <input class="tex" type="text" placeholder="Discription" [(ngModel)]="pro
jects[i].discription" >
        <input *ngIf="secetedUserr" value="Fire this user" type="button" class="b
ut" (click)="FireUser()">
        <select class="sec" *ngIf="selectedTaskk" #MySelect3 [(ngModel)]="selecte
dUserr" (change)="SelectTaskUser(MySelect3.value)" required>
            <option [ngValue]="null" [disabled]="true"> Select User </option>
            <option *ngFor="let user of projectusers" [value]="user.userId"> {{us
er.userName}} </option>
        </select> <br>
        <input class="tex" type="text" placeholder="Image" [(ngModel)]="projects[
i].image" >
        <input *ngIf="(selectedTaskk && selectedUserr)" type="button" value="Chan
ge User" class="but" (click) = "ChangeUserOnTask()"> <br>
```

```
        <input style="margin-
left: 60%;" class="but" type="button" value="Manage Users" (click)="ManageUsers()
">
        <input class="but" type="button" value="Manage Tasks" (click)="ManageTask
s()"> <br>
        <input style="margin-
left: 60%;" class="but" type="button" value="Save" (click)="Save(i)">
        <input class="but" type="button" value="Cancel" (click)="Cancel()">
    </ng-template>
</div>
```

```
import {Component, Input, OnInit} from '@angular/core'
import { User } from '../Models/User';
import { HttpClient } from '@angular/common/http'

@Component({
    selector: 'user',
    templateUrl: './user.component.html'
})
export class UserComponent implements OnInit {
    @Input()user: User;
    editing: boolean = false;
    deleted: boolean = false;
    private url='https://localhost:44312/api/User';

    constructor (private client: HttpClient) {}

    ngOnInit(): void { }

    UpdateProject():void {
        this.editing = true;
    }

    DeleteProject(): void {
        this.client.delete(this.url+'/' + this.user.userId).subscribe( (data: boo
lean) => this.deleted = data)
    }

    Save(): void {
        this.client.put(this.url + '/' + this.user.userId, this.user).subscribe((
data: User) => this.user)
    }

    Cancel():void {
        this.editing =false;
    }
}
```

```
<div style="font-size: medium; text-align: center;" *ngIf="!deleted">
    <table cellspacing="1" border = "1">
        <thead style="background-color: rgb(67, 154, 189);">
            <td> Login </td>
            <td> Password </td>
            <td> UserName </td>
            <td> Finished </td>
            <td> Failed </td>
            <td> CRUD Action </td>
        </thead>
        <tbody style="background-color: rgb(141, 191, 235);">
            <tr>
            <td *ngIf="editing"> <input type="text" [(ngModel)]="user.login" clas
s="form-control" /> </td>
            <td *ngIf="editing"> <input type="text" [(ngModel)]="user.password" c
lass="form-control" /> </td>
            <td *ngIf="editing"> <input type="text" [(ngModel)]="user.userName" c
lass="form-control" /> </td>
            <td *ngIf="editing"> <input type="number" [(ngModel)]="user.finished"
 class="form-control" /> </td>
            <td *ngIf="editing"> <input type="number" [(ngModel)]="user.failed" c
lass="form-control" /> </td>
                <td *ngIf="!editing"> {{ user.login }}  </td>
                <td *ngIf="!editing"> {{ user.password }} </td>
                <td *ngIf="!editing"> {{ user.userName }} </td>
                <td *ngIf="!editing"> {{ user.finished }} </td>
                <td *ngIf="!editing"> {{ user.failed }} </td>
            <td *ngIf = "!editing">
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Update" class="btn btn-
default" (click)="UpdateProject()" />  </p>
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Delete" class="btn btn-
default" (click)="DeleteProject()" />  </p>
            </td>
            <td *ngIf = "editing">
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Save" class="btn btn-
default" (click)="Save()" />  </p>
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Cancel" class="btn btn-
default" (click)="Cancel()" />  </p>
            </td>
            </tr>
        </tbody>
    </table>
```

```
</div>
```

```
import {Component, Input, OnInit} from '@angular/core'
import { Role } from '../Models/Role';
import { HttpClient } from '@angular/common/http'

@Component({
    selector: 'role',
    templateUrl: './role.component.html'
})
export class RoleComponent implements OnInit {
    @Input()role: Role;
    editing: boolean = false;
    deleted: boolean = false;
    private url='https://localhost:44312/api/Role';

    constructor (private client: HttpClient) {}

    ngOnInit(): void { }

    UpdateProject():void {
        this.editing = true;
    }

    DeleteProject(): void {
        this.client.delete(this.url+'/' + this.role.roleId).subscribe( (data: boo
lean) => this.deleted = data)
    }

    Save(): void {
        this.client.put(this.url + '/' + this.role.roleId, this.role).subscribe((
data: Role) => this.role)
    }

    Cancel():void {
        this.editing =false;
    }
}
```

```
<div style="font-size: medium; text-align: center;" *ngIf="!deleted">
    <table cellspacing="1" border = "1">
        <thead style="background-color: rgb(67, 154, 189);">
            <td> Role Name </td>
            <td> Role Priority </td>
            <td> CRUD Action </td>
        </thead>
        <tbody style="background-color: rgb(141, 191, 235);">
            <tr>
            <td *ngIf="editing"> <input type="text" [(ngModel)]="role.roleName" c
lass="form-control" /> </td>
            <td *ngIf="editing"> <input type="number" [(ngModel)]="role.priority"
 class="form-control" /> </td>
                <td *ngIf="!editing"> {{ role.roleName }}  </td>
                <td *ngIf="!editing"> {{ role.priority }} </td>
            <td *ngIf = "!editing">
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Update" class="btn btn-
default" (click)="UpdateProject()" />  </p>
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Delete" class="btn btn-
default" (click)="DeleteProject()" />  </p>
            </td>
            <td *ngIf = "editing">
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Save" class="btn btn-
default" (click)="Save()" />  </p>
                <p> <input style="text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236);" type="button" value="Cancel" class="btn btn-
default" (click)="Cancel()" />  </p>
            </td>
            </tr>
        </tbody>
    </table>
    </div>
```

```
import {Component, OnInit, EventEmitter, Output} from '@angular/core'
import { HttpClient } from '@angular/common/http'
import { RegisterUser } from '../Models/RegisterUser'

@Component({
    selector: 'register',
    templateUrl: './RegisterComponent.html'
})
export class RegisterComponent implements OnInit {
    @Output() resp = new EventEmitter();
    token: string;
    show: boolean = false;
    user: RegisterUser = new RegisterUser();
    private url='https://localhost:44312/api/User';

    constructor (private client: HttpClient) {}

    ngOnInit(): void { }

    async Save() {
        this.user.failed = 1;
        this.user.finished = 1;

        await this.client.post(this.url, this.user).subscribe((data: RegisterUser
) => this.user)
        this.resp.emit();
    }
}
```

```
<div style="margin-left: 32%; margin-top: 10%; background-
color: rgb(54, 90, 109); width: 35%; height: 300px;" >
    <div class="form-group">
        <input style="margin-left: 15%; width: 70%; height: 40px; margin-
top: 20px; font-
size: large;" type="text" placeholder="Create login" [(ngModel)]="user.login" cla
ss="form-control" />
    </div>
    <div class="form-group">
        <input style="margin-left: 15%; width: 70%; height: 40px; margin-
top: 20px; font-
size: large;" type="text" placeholder="Create user name" [(ngModel)]="user.userNa
me" class="form-control" />
    </div>
    <div class="form-group">
        <input style="margin-left: 15%; width: 70%; height: 40px; margin-
top: 20px; font-
size: large;" type="password" placeholder="Create password" [(ngModel)]="user.pas
sword" class="form-control" />
    </div>
    <div>
        <input style="margin-left: 72%; text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236); margin-
top: 20px;" type="button" value="Register" (click)="Save()" class="btn btn-
success" />
    </div>
</div>
```

```
import {Component, OnInit, Output,EventEmitter, Input} from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { User } from '../Models/User';

@Component({
    selector: 'profile',
    templateUrl: './profile.component.html',
    styleUrls: ['./profile.component.css']
})
export class ProfileComponent implements OnInit {
    @Output() logoff = new EventEmitter();
    @Input()  user : User;
    @Input() token : string;
    private url='https://localhost:44312/api/User';

    public isUpd: boolean = false;

    constructor (private client: HttpClient) {}

    ngOnInit(): void {  }

    async Delete() {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }
        const a: boolean = await this.client.delete<boolean>(this.url + '/' + thi
s.user.userId, requestOption).toPromise();
        this.LogOff()
    }

    async Save() {
        const headrDict = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Access-Control-Allow-Headers': 'Content-Type',
            'Authorization': 'Bearer '+ this.token
        }

        const requestOption = {
            headers: new HttpHeaders(headrDict)
        }
```

```
        await this.client.put(this.url, this.user, requestOption).subscribe((data
: User)=> this.user = data);
    }

    async LogOff() {
        this.user = null;
        localStorage.removeItem('auth_token');
        this.logoff.emit();
    }

    Update(): void {
        this.isUpd = true;
    }

    Cancel():void {
        this.isUpd = false;
    }

    Calculate():number{
        return this.user.finished/ (this.user.failed == 0 ? 0.1 : this.user.faile
d);
    }
}
```

```
<div class="prof">
    <div *ngIf="!isUpd">
        <h1> Weelcome to your Profile, {{user.userName}} </h1>
        <p> You have {{user.finished}} finished tasks </p>
        <p> and {{user.failed}} failed tasks </p>
        <p> Your rating mark now equals to {{user.rating}}  <p>
        <input style="margin-
left: 65%;" class="but" type="button" value="Logoff" (click) = "LogOff()">
        <input class="but" type="button" value="Update" (click) = "Update()">
        <input class="but" type="button" value="Delete" (click) = "Delete()">
    </div>
    <div *ngIf="isUpd">
        <input type="text" [(ngModel)]="user.login" value="user.login" class="for
m-control inp" /> <br>
        <input type="text" [(ngModel)]="user.password" value="user.password" clas
s="form-control inp" />  <br>
        <input type="text" [(ngModel)]="user.userName" value="user.userName" clas
s="form-control inp" />  <br>
        <input class="but" type="button" value="Save" (click) = "Save()">
        <input class="but" type="button" value="Cancel" (click) = "Cancel()">
    </div>
</div>
```

```
import {Component, OnInit, Output,EventEmitter} from '@angular/core'
import { HttpClient } from '@angular/common/http'
import { LoginUser } from '../Models/LoginUser';
import { analyzeAndValidateNgModules } from '@angular/compiler';

@Component({
    selector: 'login',
    templateUrl: './LoginComponent.html'
})
export class LoginComponent implements OnInit {
    @Output() resp = new EventEmitter();
    @Output() setToken = new EventEmitter();
    @Output() setUser = new EventEmitter();
    show: boolean = false;
    user: LoginUser = new LoginUser();
    private url='https://localhost:44312/api/Login';

    constructor (private client: HttpClient) {}

    ngOnInit(): void { }

    async Save() {
        const response: any = await this.client.post<any>(this.url, this.user).to
Promise();
        localStorage.setItem('auth_token', response.token);
        this.setToken.emit(response.token);
        this.resp.emit(true);
        this.setUser.emit(response.user);
    }
}
```

```
<div style="margin-left: 32%; margin-top: 10%; background-
color: rgb(54, 90, 109); width: 35%; height: 200px;" >
    <div class="form-group">
        <input style="margin-left: 15%; width: 70%; height: 40px; margin-
top: 20px; font-
size: large;" type="text" placeholder="Login" [(ngModel)]="user.login" class="for
m-control" />
    </div>
    <div class="form-group">
        <input style="margin-left: 15%; width: 70%; height: 35px; margin-
top: 20px; font-
size: large;" type="password" placeholder="Password" [(ngModel)]="user.password"
class="form-control" />
    </div>
    <div>
        <input style="margin-left: 72%; text-
align: center; width: 80px; height: 35px; background-
color: rgb(102, 182, 236); margin-
top: 20px;" type="button" value="Login" (click)="Save()" class="btn btn-
success" />
    </div>
</div>
```

```
export class LoginUser{
    constructor(
        public login?: string,
        public password?: string
    ) {}
}
export class Project{
    constructor(
        public projectId?:number,
        public title?: string,
        public status?: string,
        public image?: string,
        public discription?: string,
        public role?: string
    ) {}
}
export class ProjectUserRole{
    constructor(
        public roleId?: number,
        public projectId?: number,
        public userId?: number
    ) {}
}
export class Puttask{
    constructor(
        public taskId?: number,
        public userId?: number,
        public projectId?: number,
        public taskParentID?: number,
        public taskName?: string,
        public disription?: string,
        public status?: string,
        public createDate?: Date
    ) {}
}
export class RegisterUser{
    constructor(
        public userId?: number,
        public login?: string,
        public password?: string,
        public userName?: string,
        public finished?: number,
        public failed?: number
    ) {}
}
export class Role{
    constructor(
        public roleId?:number,
        public roleName?: string,
        public priority?: number
```

```
    ) {}
}
export class Task{
    constructor(
        public taskId?: number,
        public userId?: number,
        public userName?: string,
        public projectId?: number,
        public projectTitle?: string,
        public taskParentID?: number,
        public taskName?: string,
        public taskDiscritpion?: string,
        public taskStatus?: string,
        public taskCreateDate?: Date,
        public ratingPoints?: number,
        public taskEstimeteDate?: Date
    ) {}
}
export class User{
    constructor(
        public userId?: number,
        public login?: string,
        public password?: string,
        public userName?: string,
        public finished?: number,
        public expired?: number,
        public rating?: number,
        public failed?: number
    ) {}
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації і управління

УЗГОДЖЕНО
Керівник проєкту
_____ *Поліна*
НОВІКОВА
(підпис) (вл. ім’я, прізвище)
“13” квітня 2020 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр*
ПАВЛОВ
(підпис) (вл. ім’я, прізвище)
“14” квітня 2020 р.

Інформаційна система технології менеджменту проектів з використанням Rest API

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП 6326.01.000 ТЗ*

на 11 сторінках

Київ – 2020 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Інформаційна система технології менеджменту проектів з використанням Rest API

Коротке найменування системи: «Project Management».

1.2 Найменування організації-замовника та організації-учасника робіт

Замовником є кафедра автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" (далі за текстом — Замовник). Адреса замовника: м. Київ, п. Перемоги 37, 18 корпус ФІОТ.

Розробники сервісу — студенти групи ІС-63 кафедри автоматизованих систем обробки інформації та управління Національного технічного університету України "Київський політехнічний інститут ім. Ігоря Сікорського" Фомін Ілля Дмитрович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над створенням інформаційної системи технології менеджменту проектів – 1 грудня 2019 року.

Плановий термін по закінченню роботи над створенням інформаційної системи технології менеджменту проектів – 22 травня 2020 року.

2 ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

2.1 Призначення системи

Інформаційна система контролю створення та управління проектами призначена для спрощення контролю виконання поставлених на проекті задач та завдань також для покращення звітності виконаної роботи.

2.2 Цілі створення системи

Основний показник ефективності який збільшує дана інформаційна система – це контроль виконання або управління проектами. Ефективність підприємств, що будуть систему контролю та управління проектами однозначно збільшиться, адже голова проектів буде мати повну звітність щодо виконаної роботи.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для того, щоб користуватися системою, користувач повинен бути зареєстрованим та авторизованим. Для клієнтів необхідно створити приватний акаунт.

Об'єктами автоматизації є збір інформації щодо виконаних завдань на проектах користувачів та виставлення їм оцінок.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Користувачам необхідно надавати відповідний до їх ролі на проекті можливості. Так, актор “Голова проекту” може видаляти, створювати завдання на проекті, переназначати виконавців завдань, змінювати дані проекту, видаляти учасників з проекту, змінювати статус як завдань так і проектів.

Для виконання даної вимоги необхідно ввести в систему етап авторизації, який буде ідентифікувати користувача, що намагається виконати певні дії на проекті або з завданням. Якщо, роль користувача на проекті не дозволяє виконати певну дію з проектом або завданням тоді, ці дії навіть не повині демонструватися користувачу у програмному забеспечені.

4.2 Вимоги до надійності

Система повинна функціонувати незважаючи на наявність ймовірних дефектів та не коректних вводів від користувачів, які можуть проявлятися під час експлуатації. Такі вводи та дефекти повинні запобігатись на етапі взаємодії користувача з системою, тобто вказувати на некоректні вводи та не давати змоги користувачу виконати дію яку він намагався виконати з некоректними даними. У разі виконання некоректної дії користувачем ПО на кожному етапі взаємодії повинно перевіряти інформації повідомлення.

Жодна з некоректних дій користувача не повинна призводити до краху системи.

Будь-які аварійні ситуації повинні бути оброблені та видавати повідомлення про некоректність дій або даних користувача.

4.3 Вимоги до складу і параметрів технічних засобів

Мінімальні вимоги для технічного забезпечення клієнту:

RAM: 512Мб,

Місце на диску: 200Мб,

Процесор: одно-ядерний, 1,2ГГц,

Відеокарта: 512Мб,

Монітор: будь-який,

Браузер: Google Chrome

Рекомендовані вимоги для технічного забезпечення клієнту:

RAM: 1024Мб,

Місце на диску: 200Мб,

Процесор: одно-ядерний, 1,5ГГц,

Відеокарта: 512Мб,

Монітор: будь-який,

Браузер: Google Chrome

Мінімальні вимоги для технічного забезпечення серверу:

RAM: 1024Мб,

Місце на диску: 1200Мб,

Процесор: одно-ядерний, 1,5ГГц

Рекомендовані вимоги для технічного забезпечення серверу:

RAM: 1024Мб,

Місце на диску: 1200Мб,

Процесор: одно-ядерний, 1,5ГГц

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з розробки інформаційної системи технології менеджменту проектів:

№ з/п	Назва етапу роботи	Термін виконання етапу	Результат виконання
1.	Підготовка технічного завдання на розробку програмного продукту	02.12.20119	
2.	Розробка сценарію роботи	14.12.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	26.12.2019	
4.	Узгодження з керівником інтерфейсу користувача	11.01.2020	
5.	Розробка інформаційного забезпечення	21.01.2020	
6.	Розробка програмного забезпечення	01.02.2020	
7.	Налагодження програми	09.04.2020	
8.	Тестування програми	19.04.2020	
9.	Здача готового програмного продукту замовнику	24.05.2020	

6 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

6.1 Види випробувань

З метою перевірити правильності роботи програмного забезпечення необхідно буде провести функціональне тестування, в ході якого буде виконано перевірку всіх функціональних характеристик застосунку на правильність виконання. Система буде перевірена на стійкість до незапланованих дій користувача. Система безпеки буде перевірена окремими тестами.

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

ID перевірки:
1003783780

Дата перевірки:
04.06.2020 18:33:57 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2020 20:48:30 EEST

ID користувача:
77149

Назва документу: Fomin_bachelor

ID файлу: 1003798192 Кількість сторінок: 47 Кількість слів: 5727 Кількість символів: 40932 Розмір файлу: 2.30 MB

5.71% Схожість

Найбільша схожість: 3.6% з джерело бібліотеки. ID файлу: 5857455

3.39% Схожість з Інтернет джерелами 17 Рядів 49

5.71% Текстові збіги по Бібліотеці акаунту 135 Рядів 49

6.02% Цитат

Цитати 6 Рядів 50

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

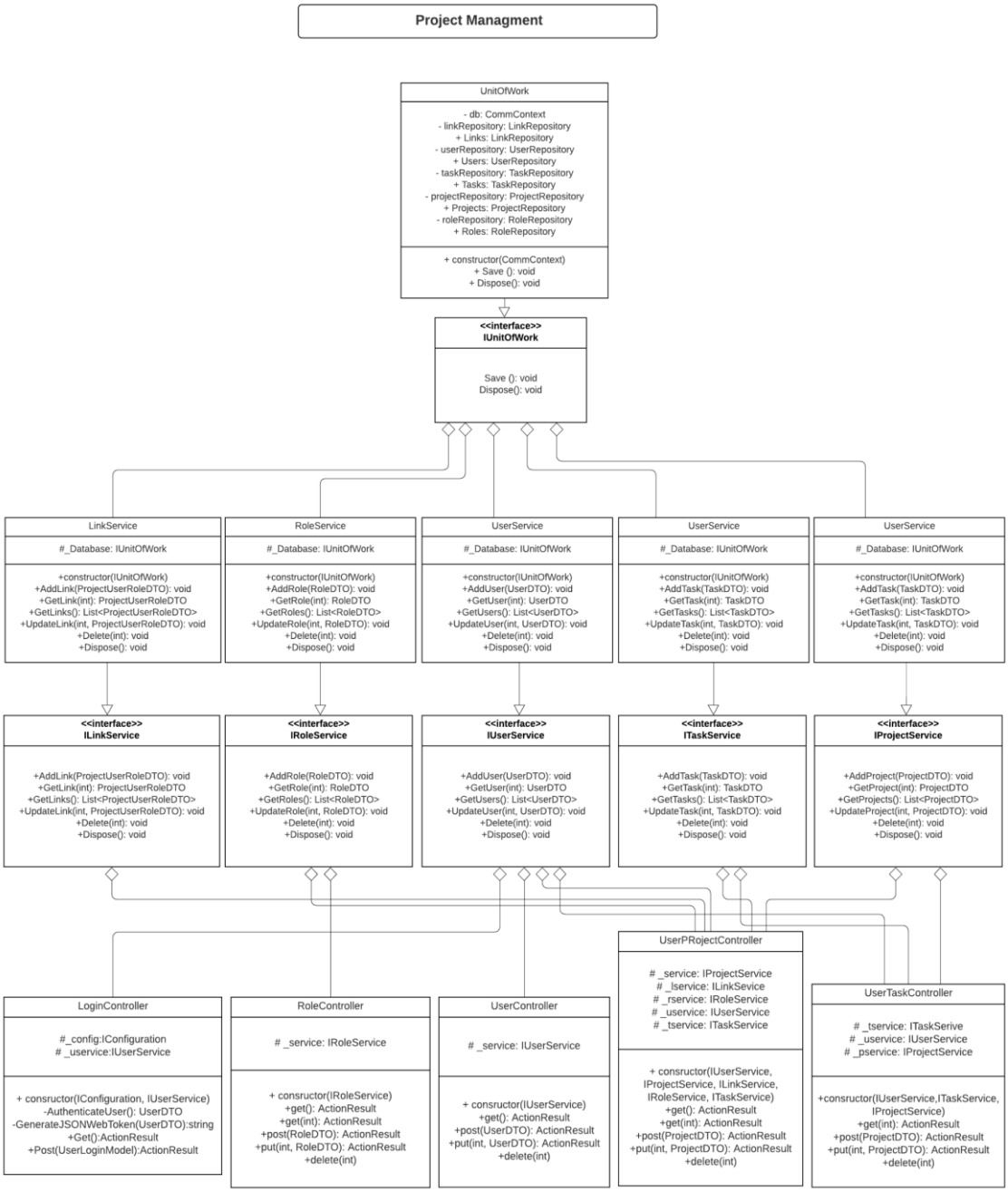
Не знайдено заміненних символів

Графічний матеріал
до дипломного проєкту

на тему: Інформаційна система технології менеджменту проектів з використанням Rest API

Київ – 2020 року

					ДП 6326.02.000.ССВ			
Зм.	Арк.	№ документа	Підпис	Дата		Літера	Маса	Масштаб
Розробив		Фомін І.Д.			Схема структурна варіантів використання			
Перевірив		Новікова П.А.				Аркуш 1		Аркушів 1
Т. кон.								
					Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41		
Н. кон.		Проскура С.Л.						
Затвердив		Новікова П.А.						

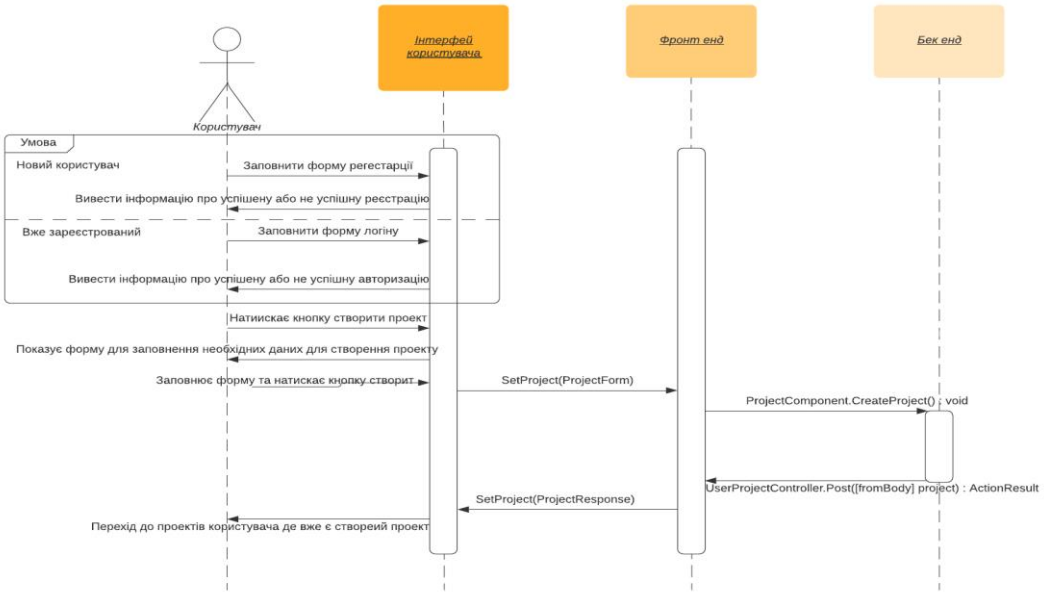


					ДП 6326.03.000.ССК	
Зм.	Арк.	№ докум.	Підп.	Дата		
Розроб.		Фомін І.Д.			Схема структурна класів програмного забеспечення	Лит. / Маса / Масштаб
Перев.		Новікова П.А.				
Т. Кон.						Аркуш 1 / Аркушів 1
Н. Кон.		Проскура С.Л.			Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41
Затв.		Новікова П.А.				

ДП 6326.04.000.СБД

					ДП 6326.04.000.СБД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема баз даних	Літера / Маса / Масштаб
Розробив		Фомін І.Д.				
Перевірив		Новікова П.А.				Аркуш 1 / Аркушів 1
Т. кон.						
Н. кон.		Проскура С.Л.			Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41
Затвердив		Новікова П.А.				

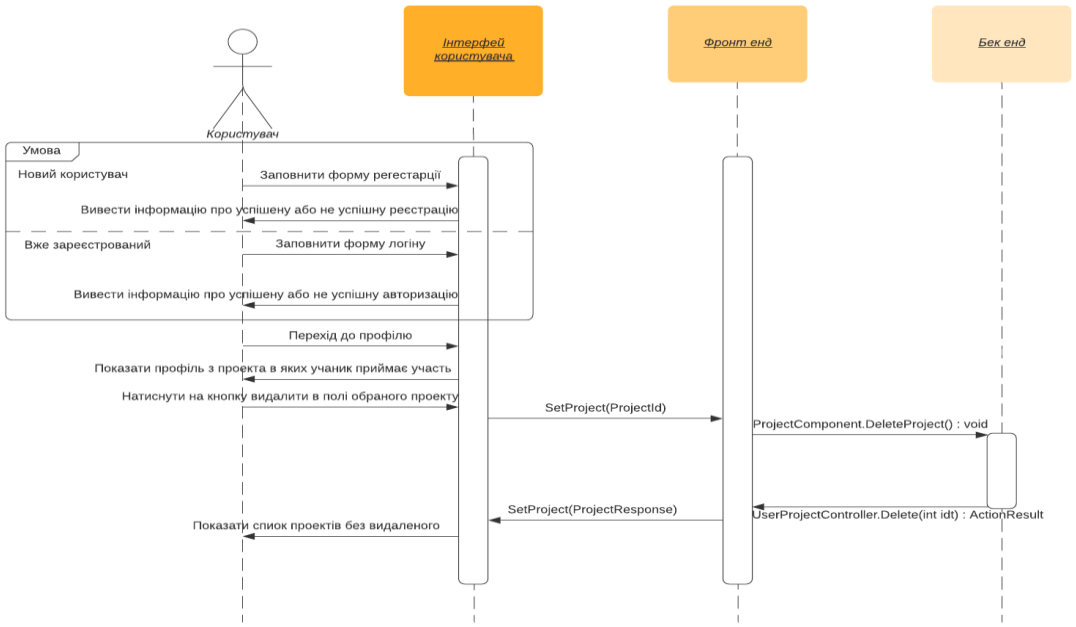
ДП 6326.05.000.ССП



					ДП 6326.04.000.ССП	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера / Маса / Масштаб
Розробив		Фомін І.Д.				
Перевірив		Новікова П.А.				
Т. кон.						Аркуш 1 / Аркушів 1
Н. кон.		Проскура С.Л.			Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41
Затвердив		Новікова П.А.				

ДП 6326.06.000.ССП

					ДП 6326.06.000.ССП		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності	Літера	Маса / Масштаб
Розробив		Фомін І.Д.					
Перевірив		Новікова П.А.				Аркуш 1	Аркушів 1
Т. кон.							
Н. кон.		Проскура С.Л.			Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41	
Затвердив		Новікова П.А.					



					ДП 6326.07.000.ССП			
						Літера	Маса	Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовності			
Розробив		Фомін І.Д.				Аркуш 1	Аркушів 1	
Перевірив		Новікова П.А.						
Т. кон.								
					Інформаційна система технології менеджменту проектів з використанням Rest API	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-41		
Н. кон.		Проскура С.Л.						
Затвердив		Новікова П.А.						